

Verifiable Computation on Encrypted Data from Homomorphic Encryption over Random Encodings

Jiayi Kang^{1*}, Erik Pohle², and Jannik Spiessens³

¹ Faculty of Information Technology, Monash University, Australia
jiayi.kang@monash.edu

² Aarhus University, Aarhus, Denmark
erik.pohle@cs.au.dk

³ COSIC, KU Leuven, Leuven, Belgium
jannik.spiessens@esat.kuleuven.be

Abstract. Verifiable Computation on Encrypted Data (vCOED) enables clients to outsource computation while providing *both* privacy and integrity. A promising vCOED paradigm is blind proofs, which use homomorphic encryption (HE) to evaluate verifiable computation (VC) proofs on encrypted data. Despite their efficiency, blind proofs do not provide *strong* client privacy: a malicious server can exploit reaction attacks to learn information about the client’s inputs, intermediate values or final outputs.

Our work addresses this limitation by introducing *Homomorphic Encryption over Random Encodings* (HERE). We formalize a new security notion, indistinguishability under chosen-plaintext attacks in the presence of an equality oracle over encodings (IND-CPA-E). We prove that combining IND-CPA-E-secure HERE schemes with blind proofs that verify computation over the encodings yields vCOED protocols *with* strong client privacy. We then instantiate a HERE scheme for functions of bounded degree d , using Shamir secret sharing at $(d + 1)$ evaluation points under $(d + 1)$ distinct HE keys. Finally, we demonstrate the concrete efficiency of our approach through an application to matrix-vector multiplication (MVM), a linear computation corresponding to $d = 1$. The proof overhead (i.e. the ratio of the vCOED prover runtime to the homomorphic MVM runtime) in our combined vCOED protocol is as low as $2\times$, as demonstrated in our prototype implementation.

Keywords: Verifiable Computation on Encrypted Data · Homomorphic Encryption · Matrix-Vector Multiplications.

1 Introduction

Cloud services have become a popular paradigm for data storage and computation. However, this also introduces two challenges: *privacy*, i.e. whether the cloud

* Work partially done while the author was at COSIC, KU Leuven.

can perform computations without learning clients' private data, and *integrity*, i.e. whether a client can efficiently verify that the server performs the correct computation.

Verifiable Computation on Encrypted Data (vCOED), as introduced in [31], addresses both challenges simultaneously. More precisely, when outsourcing the computation $y = C(x)$, the client sends only encrypted inputs $\text{Enc}(x)$ and receives encrypted outputs $\text{Enc}(y)$ as well as a proof from the server. If the proof verifies, then the output corresponds to the desired computation result $C(x)$; otherwise, the client aborts. A vCOED scheme should satisfy three key properties:

- Correctness. An honest server always produces a proof accepted by the client, who obtains the correct result $y = C(x)$ upon decryption.
- Soundness⁴. No malicious server can convince the client to accept an incorrect output, except with negligible probability.
- Strong client privacy. A malicious server learns nothing about the client's inputs (and therefore also intermediate values and final outputs), even if it observes the client's verification outcome (accept or reject).

Initiated by Gennaro et al. [38], substantial research has explored the construction of *non-interactive* vCOED schemes by combining homomorphic encryption (HE) with verifiable computation (VC). A prominent direction is the *verifiable homomorphic encryption* (vHE) paradigm, where the server generates a proof for the correct execution of homomorphic evaluation, i.e. the proof is produced over HE *ciphertexts*. This was first explored using ad-hoc VC protocols for restricted function classes, such as linear functions or degree-bounded polynomials [31]. Later works mostly focus on general circuits and succinct proof systems such as SNARKs [32, 52, 3, 72, 67, 57].

Despite this progress, proving *ciphertext* relations remains prohibitively expensive in practice. First, ciphertexts have large dimensions and ciphertext operations, such as key-switching and bootstrapping, are inherently non-arithmetic and therefore costly to prove. Second, practical HE implementations often employ optimisations such as ring-switching and modulus-switching to *reduce* computation and communication overheads. In the vHE setting, however, these optimisations must also be incorporated into the proof, further *increasing* the overall cost.

This naturally raises the following question: can we construct vCOED protocols that prove the (encoded) *plaintext* relation $y = C(x)$, rather than proving its correct execution on *ciphertexts*, thereby decoupling the proof from the underlying HE evaluation process? Ideally, such vCOED schemes with plaintext verifiability should provide correctness, soundness, and strong client privacy. As correctness follows directly from the correctness of the underlying VC and HE components, we focus on the latter two properties.

⁴ For vCOED protocols where the server holds private inputs, [76] shows that knowledge soundness is necessary. We focus on delegation settings where server's inputs are public and incorporated into the description of the function C , hence soundness suffices.

Regarding soundness, the work [17] attempts to achieve plaintext verifiability by encoding precomputed challenge values together with messages through specially designed plaintext encoders. However, as subsequently shown in [21], such vCOED constructions do not provide soundness. A different approach to achieve plaintext verifiability is offered by blind proofs [36, 2, 34, 76, 67, 77]. In this paradigm, a VC protocol for the plaintext relation is evaluated homomorphically, and the resulting proof is verified in the plaintext domain after decryption. The soundness of blind proofs is formally proven in [34], which also shows a performance improvement of four orders of magnitude compared to vHE. The efficiency of blind proofs has been further enhanced in subsequent works [76, 67, 77] by exploiting HE-friendly VC constructions, bringing blind proofs closer to practical performance.

Nevertheless, blind proof constructions do not achieve strong client privacy. Reaction attacks [22, 78] exploit the server’s capacity to observe the client’s reaction after decrypting tampered ciphertexts. This attack for homomorphic encryption already exists at the system level: a client can notice invalid decryption results, e.g. when the decrypted text looks gibberish, and react in a detectably “unexpected” way. Blind proofs make this detection explicit by providing a concrete primitive for plaintext verification, allowing the client to reliably determine whether decrypted results satisfy the expected relation. In other words, the server (adversary) treats the client as a verification oracle that outputs a bit indicating whether the plaintext was altered. We capture this capacity by extending the classic indistinguishability against chosen plaintext attacks (IND-CPA) game with an equality test oracle, and call the resulting notion IND-CPA-E.

An example reaction attack is the one-bit forge attack [78, 76]. To determine whether a ciphertext $\text{Enc}(b)$ encrypts $b = 0$ or 1 , a malicious server replaces it with $\text{Enc}(0)$ before continuing the protocol. If the resulting response passes verification, the server learns that $b = 0$; otherwise $b = 1$. Although each verification outcome leaks at most one bit [76], this leakage can be significant when the client’s input, intermediate values, or the output have low entropy. For example, unit vectors commonly arise as input values in private information retrieval (PIR) [1, 46, 30] and intermediate values in private decision-tree evaluation (PDTE) [24, 62, 25]; then a single leaked bit can reveal whether a particular index (or leaf) was queried or returned, or whether the queried index (or returned leaf) lies within a chosen subset of the records.

Our work augments blind proofs to achieve strong client privacy. The key idea is to randomize the computation using *random homomorphic correlations*, which we formalize through a new primitive called *Homomorphic Encryption over Random Encodings (HERE)*. More specifically, to encode an input x , the client samples a random correlation (that would reconstruct to x) which is encrypted and then sent to the server. The server evaluates the desired function homomorphically by evaluating the corresponding homomorphic operation over the correlation, and also uses blind proofs to provide soundness. Note that client reaction can be exploited to infer information about the random correlation, but

not the underlying value x . We give a more detailed overview of the formalisation and construction later in Section 1.2. We first summarize our contributions.

1.1 Contributions

Our contributions are three-fold.

- We introduce Homomorphic Encryption over Random Encodings (HERE), which formalizes the notion of encrypting random correlations. We define an appropriate security notion, namely IND-CPA security where the adversary has one-time access to an equality oracle over decryptions, which we call IND-CPA-E. We show that combining an IND-CPA-E secure HERE scheme with blind proofs yields a vCOED protocol with strong client privacy. This is detailed in Section 3.
- We construct a HERE scheme for bounded-degree functions by combining homomorphic encryption with Shamir’s secret sharing scheme [70]. To evaluate a degree- d function, the scheme encodes each message into $(d + 1)$ components, each encrypted under a distinct HE key. Section 4 details the construction. We prove that the resulting instantiation is IND-CPA-E secure under a multi-key and bounded-depth circuit-awareness assumption. Our notion of circuit awareness generalises plaintext awareness [7, 5, 6] and linear-function awareness [9, 48], and may be of independent interest.
- We build vCOED protocols using the approach described above, for matrix-vector multiplication (with $d = 1$). Our first protocol, vCOEDMVI, is only $2\times$ slower than a single homomorphic multiplication between a plaintext matrix and a ciphertext vector (PC-MV), though its offline setup is expensive for the client. We further propose a sumcheck-based protocol, vCOEDMVII, to securely delegate the setup. The resulting combined protocols are described in Section 5 and detailed in Appendix C. Our Rust prototype achieves up to 45.2 MB/s on a single core. In an indicative comparison with the reported 14-thread results of [73] for matrices of similar sizes, our protocol exhibits at least $35\times$ lower proof overhead and $400 - 570\times$ smaller response sizes, although the two systems use different plaintext fields and matrix dimensions.

1.2 Technical Overview

We continue the discussion of our techniques.

HERE for strong client privacy. A HERE scheme differs from a homomorphic encryption scheme by two additional algorithms `Encode` and `Decode`. A message x is encoded into $c_x \leftarrow \text{Encode}(\text{sk}_1, x)$. The encoding c_x is further encrypted, yielding ciphertext $\text{ct}_x \leftarrow \text{Encrypt}(\text{sk}_2, c_x)$. The server homomorphically evaluates a function f on ct_x to obtain ciphertext $\text{ct}_{f(x)}$. To recover the computation result, $\text{ct}_{f(x)}$ is first decrypted into $c_{f(x)}$, which is then decoded as $f(x) := \text{Decode}(\text{sk}_1, c_{f(x)})$.

Security is defined as indistinguishability under chosen plaintext attack with an equality oracle (IND-CPA-E) which augments the standard IND-CPA security game for homomorphic encryption with the oracle $\mathcal{O}_{\text{Equal}}$, returning whether two ciphertexts decrypt to the same encoding.

Strong client privacy extends client privacy by giving the server access to the client’s verification output on (potentially adversarially chosen) ciphertexts. When a HERE scheme is combined with blind proofs, this verification output reveals whether the decrypted results satisfy an expected relation on the encodings. This leakage is modelled in the IND-CPA-E security game. We prove that combining an IND-CPA-E secure HERE with blind proofs yields a vCOED protocol with strong client privacy in Theorem 2 (Section 3).

HERE scheme for bounded-degree functions. A *heavyweight* HERE construction uses HE as the randomised encoding, resulting in a two-layer HE structure.⁵ Although this construction establishes the feasibility of HERE for arbitrary functions, it is not concretely efficient. On the other hand, *hiding* the underlying values as done by the inner HE scheme is not a necessary property for a secure HERE encoding. It suffices to *randomize* the computation so that the output of the equality oracle $\mathcal{O}_{\text{Equal}}$ does not reveal additional information on the underlying messages. With this in mind, we propose a *lightweight* HERE scheme for bounded-degree functions.

Warm-up: An insecure correlation/encoding. We encode x as random $(c_{x,1}, c_{x,2})$ subject to $c_{x,1} + \alpha \cdot c_{x,2} = x$, where α is the secret encoding key sk_1 . Then, both components are encrypted using the HE secret key $\text{sk}_2 = \text{sk}_{\mathcal{E}}$. Since the correlation is linearly homomorphic, it would be a candidate for linear functions. Unfortunately, the resulting HERE scheme cannot satisfy the IND-CPA-E notion. Specifically, the adversary chooses two messages m_0 and m_1 , and receives $\text{ct}_1^*, \text{ct}_2^*$ from the IND-CPA-E challenger, where ct_1^* and ct_2^* are encryptions of c_1^* and c_2^* , respectively, s.t. $c_1^* + \alpha \cdot c_2^* = m_b$ for the secret, random bit b of the challenger. Let $c_1 + \alpha \cdot c_2 = m_0$ be an encoding of m_0 that the adversary obtained, e.g. via an encoding oracle. Observe that the adversary can perform the following *mix cross-terms* attack, since computing

$$c_1^* \cdot c_2 - c_1 \cdot c_2^* = c_2 \cdot m_b - c_2^* \cdot m_0, \quad (1)$$

homomorphically over the respective ciphertexts yields an expression with α cancelled out! Thus, by choosing $m_0 = 0$ and $m_1 \neq 0$, and testing whether Eq. (1) equals 0 via the equality oracle, the adversary can conclude whether $b = 0$.

Our encoding for bounded-degree functions. In our main construction, a HERE scheme for bounded-degree functions that can be expressed as multivariate polynomials of degree at most d (where d is a tunable parameter), we prevent cross-term mixing and generalize to functions with degree larger than one.

⁵ A similar construction appears in [68], where LWE serves as the inner HE scheme and RLWE as the outer HE scheme to verifiably compute PIR hints.

To prevent cross-term mixing, we encrypt each component of the encoding (in the previous example: $c_{x,1}$ and $c_{x,2}$) with a different HE key. This separates the components and renders attempts to mix ciphertexts from different keys futile. We require wrong-key decryption [58, 33] as an additional property of the HE scheme.

As a correlation that supports degree- d multivariate polynomials, we employ Shamir’s secret sharing scheme [70]. To encode the value x , pick a random degree-1 polynomial p subject to $p(0) = x$. The encoding consists of $d + 1$ components, where each component is p evaluated at a public, distinct, non-zero evaluation point α_i , i.e. $c_{i,x} = p(\alpha_i)$. Therefore in our construction, $\mathbf{sk}_1$ is empty. The final $d + 1$ ciphertexts consist of each $c_{i,x}$ encrypted under the i -th secret key. The correlation is linearly homomorphic and multiplicatively homomorphic for bounded-degree functions. Since we supply $d + 1$ components, which are evaluations at distinct points, a client can reconstruct the final polynomial as long as the degree of the outsourced function is $\leq d$.

In Theorem 3 (Section 4), we show that this encoding indeed satisfies IND-CPA-E security assuming (i) the adversary is multi-key, L -bounded circuit-aware, and (ii) the levelled HE scheme supports large HE plaintexts of $(\lambda + L)$ -bit, where λ is the security parameter and L is the maximum number of consecutive ciphertext-ciphertext multiplications. Our definition of bounded-depth circuit awareness is a natural extension of plaintext awareness [7, 5, 6] and linear-function awareness [9, 48]; see Section 1.3 for a detailed discussion. The multi-key variant models wrong-key decryption, where the adversary passes ciphertexts encrypted under different keys to the verification/equality oracle. The resulting construction can be summarized in the following informal result.

Theorem (informal). *Let C be a multivariate polynomial of degree at most d over the finite field $\mathbb{F}$, $\mathcal{E}$ be a levelled homomorphic encryption scheme supporting L consecutive ciphertext-ciphertext multiplications and $\mathcal{VC}$ a verifiable computation scheme for C . Let $\mathbf{vCOED}^{\mathcal{BP}}$ denote the blind proof vCOED scheme for C using $\mathcal{E}$ and $\mathcal{VC}$ (which does not have strong client privacy). Then, under the L -bounded circuit awareness assumption, there exists a vCOED scheme with strong client privacy for C with the following properties:*

- *it has key size $(d + 1)$ times the key size of $\mathbf{vCOED}^{\mathcal{BP}}$,*
- *its query, server computation and verification cost are $(d + 1)$ times the cost of $\mathbf{vCOED}^{\mathcal{BP}}$, and*
- *it has a strong client privacy leakage error of $2^L/|\mathbb{F}| + \text{negl}(\lambda)$.*

Application: vCOED for matrix-vector multiplication. As an application, we consider a client who holds a secret vector $\vec{u} \in \mathbb{F}^n$ and wants to obtain the matrix-vector multiplication (MVM) $\vec{v} = \mathbf{A}\vec{u} \in \mathbb{F}^m$, for a large public matrix $\mathbf{A} \in \mathbb{F}^{m \times n}$. In secure delegation using homomorphic encryption, the server homomorphically computes $\text{Enc}(\vec{v}) = \mathbf{A} \cdot \text{Enc}(\vec{u})$ using a PC-MV. Our goal is to additionally provide integrity, thereby transforming PC-MV into a vCOED protocol for MVM. We achieve this by combining our HERE instantiation for

linear functions (i.e. $d = 1$), with blind proofs constructed from suitable VC protocols.

The vCOEDMVI protocol. We first use a classic VC protocol based on random linear checks [11, 12]. In the offline setup phase of this vCOEDMVI protocol, the client samples a random row vector $\mathbf{r} \leftarrow \mathbb{F}^m$, computes $\mathbf{s} = \mathbf{r} \cdot \mathbf{A} \in \mathbb{F}^n$, and stores $(\mathbf{r}, \mathbf{s})$ as the verification key. For any input $\vec{u}$, the client encodes it into two vectors $\vec{c}_{u,1}, \vec{c}_{u,2} \in \mathbb{F}^n$, and sends the encryptions under distinct keys $\text{ct}[\vec{c}_{u,i}] \leftarrow \mathcal{E}.\text{Enc}_{\text{sk}_{\mathcal{E},i}}(\vec{c}_{u,i})$, for $i = 1, 2$, to the server. The server homomorphically computes $\text{ct}[\vec{c}_{v,i}] = \mathbf{A} \cdot \text{ct}[\vec{c}_{u,i}]$ using the corresponding public evaluation keys. Upon receiving the resulting ciphertexts, the client decrypts each $\text{ct}[\vec{c}_{v,i}]$ with the corresponding secret key $\text{sk}_{\mathcal{E},i}$ to recover $\vec{c}_{v,i}$. The client then verifies their correctness by checking $\mathbf{s} \cdot \vec{c}_{u,i} \stackrel{?}{=} \mathbf{r} \cdot \vec{c}_{v,i}$, for $i = 1, 2$. If both checks pass, the client reconstructs $\vec{v}$ from $\vec{c}_{v,1}$ and $\vec{c}_{v,2}$. Note that this vCOED protocol is only $2\times$ slower than a single PC-MV, as the server now performs two PC-MV evaluations.

vHE protocol from the same VC. We next explain why constructing a vHE-based vCOED protocol as in [75] from this classic random linear combination is considerably more restrictive. Note in [75], homomorphic evaluation only consists of linear ciphertext operations. As entries in the vector $\vec{u}$ are elements in the field $\mathbb{F}$, a common optimisation [71] is to encode multiple entries in $\vec{u}$ into one ciphertext, so that subsequent computations can be computed in a Single-Instruction Multiple-Data (SIMD) manner. Then PC-MV requires operations such as automorphism-based rotations, which are no longer linear and are known to be particularly expensive to prove [3]. By contrast, our HERE-based blind-proof approach decouples the HE evaluation from the relation being proved. This comparison illustrates a core advantage of our approach: verifying relations among *encoded plaintexts* is simpler and more flexible than directly verifying operations on HE *ciphertexts*.

vCOEDMVII and our combined protocol. The main limitation of vCOEDMVI is its expensive offline setup, which includes one MVM $\mathbf{s} = \mathbf{r} \cdot \mathbf{A}$ for a random vector $\mathbf{r}$. To ensure soundness, the verification key $\text{vk} = (\mathbf{r}, \mathbf{s})$ must remain hidden from the server. It should therefore be generated either by a trusted third party that reveals it only to the client, or by the client itself. In the latter case, although the one-time offline setup cost can be amortized over multiple queries (without increasing soundness error), the setup costs may still be prohibitively expensive for resource-constrained clients.

To make setup costs practical, we introduce another protocol, vCOEDMVII, that uses a sumcheck-based proof system, which is similar to the lincheck step of Spartan [69] except it is tailored to MVMs. This protocol does not require expensive verifier preprocessing. Instead, the publisher of $\mathbf{A}$ can include a commitment to $\mathbf{A}$, which can be used by different clients and allows for public verification. However, this protocol incurs larger proof overhead, with server computation of $\mathcal{O}(mn \log(mn))$ compared to $\mathcal{O}(mn)$ in vCOEDMVI for $\mathbf{A}$ of size $m \times n$.

We further combine the two protocols to obtain the advantages of both. In the offline setup phase, we use vCOEDMVII to allow the client to securely and ver-

ifiably delegate the computation $\mathbf{s} = \mathbf{r} \cdot \mathbf{A}$. Crucially, the derived verification key $\mathbf{vk}$ remains secret from the server. The online phase is identical to vCOEDMVI, maintaining $\mathcal{O}(mn)$ server computation cost.

Implementation and results. We implement our three vCOED protocols using the GBFV scheme [37] provided by the Fheanor library [64]. Across all evaluated matrix sizes, our combined protocol has a $2\times$ proof overhead (the ratio of Compute time in vCOED to a single PC-MV runtime) and achieves a throughput of up to 45.2 MB/s using a single core. The vHE-based vCOED protocol for MVM in [73] has proof overheads of $71 - 90\times$ and throughput below 2 MB/s using 14 threads. Comparing the reported results for similar matrix sizes, our protocol exhibits $400 - 570\times$ smaller responses (server-to-client communication including the proof)⁶. The comparison is presented in Table 2 and visualized in Fig. 5.

Organization. This concludes the overview. In Section 1.3, we discuss related work in the literature. We give preliminary information in Section 2 and then continue with the definition of HERE in Section 3. This is followed by our instantiation for bounded-degree functions in Section 4. Then, we discuss the application to verifiable matrix-vector products in Section 5 and conclude in Section 6.

1.3 Related works

Security notions for HE. A related line of work augments IND-CPA security into IND-CCA-1 [60] and IND-CPA-D [55, 18, 20] in the context of HE, where the adversary has access to a decryption oracle under certain constraints. On the one hand, the decryption oracle is more powerful than our verification/equality oracle, as it returns the full decryption result rather than a single bit. On the other hand, an IND-CCA-1 adversary has access to the decryption oracle only before seeing the challenge ciphertext, and an IND-CPA-D adversary can query decryptions only for well-formed input ciphertexts whose underlying plaintexts are independent of the challenge bit. In our IND-CPA-E game, the adversary has access to the verification/equality oracle after receiving the challenge, and the underlying plaintexts of input ciphertexts can depend on the challenge bit. Consequently, IND-CCA-1/IND-CPA-D secure HE is not automatically IND-CPA-E secure, which motivates our construction of HERE.

Bounded-depth circuit awareness. When instantiating HERE in Section 4, we introduce a notion of bounded-depth circuit awareness. Our definition is inspired by plaintext awareness (PA) [7, 5, 6] for (non-homomorphic) public-key encryption schemes. Intuitively, an encryption scheme is PA if the only way that an adversary can produce a valid ciphertext is to apply the encryption algorithm to the public key and a message.

⁶ Unfortunately, the concrete response size in [73] is larger than the size of $\mathbf{A}$ since each output ciphertext does not encode multiple elements, which rules it out as a candidate for our offline setup in the combined protocol.

The work [9] extends this notion of PA to allow linear evaluation for additive HE, i.e. whenever the adversary produces a valid ciphertext, it is possible to efficiently extract a corresponding affine function “explaining” the ciphertext. Our work further extends it to bounded-depth circuit evaluation: an adversary is *bounded-depth circuit-aware* if every ciphertext it produces decrypts to a bounded-depth circuit output of the plaintexts of existing ciphertexts. We call such ciphertexts *well-formed*, and only well-formed ciphertexts can be submitted to the verification/equality oracle. This definition mirrors the database notion in the IND-CPA-D game, which records all ciphertexts formed by using the encryption and evaluation oracles. Input ciphertexts to the decryption oracle in the IND-CPA-D game must appear in this database and are challenge-independent.

Bounded-depth circuit awareness is a strong assumption on the adversary behaviour. To the best of our knowledge, however, the only known attacks that exploit ciphertexts outside this model are key-recovery attacks [78, 19] through repeated malformed ciphertext queries. In our one-time verification model, such an attack can reveal at most one bit of the secret key. Unlike the direct leakage about client data addressed in this work, such leakage does not immediately disclose sensitive application-level information. Moreover, its concrete security impact can be estimated using the Leaky Estimator [27, 28], as suggested in [2]. We discuss these security implications and HE key reusability in Remark 3.

Homomorphic encrypted authenticators. The notion of homomorphic authenticators by Gennaro and Wichs [39] is also related to verifiable computation. The tag polynomial in the construction of [16] is closely related to our HERE encoding in Section 4. In [16], the authenticator of the i -th message m_i are the coefficients y_0, y_1 of the degree-1 polynomial that evaluates $y(0) = m_i$ and $y(\Delta) = F_K(i)$, where F is a PRF, and Δ and K are secret verification keys. The polynomial size in both our HERE encoding and in [16] grows linearly with the degree of the circuit being evaluated.

Replacing our HERE encoding with the construction in [16], as done in [17], reveals two crucial differences. First, regarding *soundness*: the soundness of our construction relies on the soundness of blind proofs, with the encoding only serving to provide strong client privacy, whereas [17] relies solely on the evaluation of the homomorphic authenticator for soundness, which was later attacked in [21]. Second, regarding *strong client privacy*: since $y_0 = m_i$, the tag directly leaks the message. The resulting construction would not satisfy the security requirements of a HERE scheme as an encryption of $y_0, \text{ct}[y_0]$, leaks m_i under the IND-CPA-E equality oracle. Other works [54, 56] similarly built vCOED protocols using homomorphic encrypted authenticators. These works rely on pairing-based assumptions (hence not post-quantum secure), and do not provide strong client privacy. Combining their constructions with an IND-CPA-E HERE scheme would additionally provide strong client privacy.

2 Preliminaries

Notation. We use $\mathbb{F}$ to denote a finite field and λ to denote the security parameter. We use $\text{negl}(\lambda)$ to denote an arbitrary negligible function, i.e. any function f such that $f(\lambda) = o(\lambda^{-c})$ for any $c \in \mathbb{N}$. We write $a \leftarrow \$ A$ to sample uniformly from the finite set A . For an integer n , we write the set $[n] = \{1, 2, \dots, n\}$. We use $\text{ct}[x]$ to denote HE encryptions of x , and ct_x to denote HERE encryptions of x . For some ring R , we use $R^{\leq d}[X_1, \dots, X_\mu]$ to denote the set of μ -variate polynomials where the degree of each variable is at most d .

2.1 Homomorphic encryption

Definition 1 (Homomorphic encryption). An HE scheme $\mathcal{E} = (\text{KeyGen}, \text{Enc}, \text{Dec}, \text{Eval})$ with plaintext space $\mathcal{P}$ and ciphertext space $\mathcal{C}$ includes the following polynomial-time algorithms:

- $\mathcal{E}.\text{KeyGen}(1^\lambda) \rightarrow (\text{sk}, \text{pk})$ takes the security parameter λ , outputs a secret key sk and a public key pk .
- $\mathcal{E}.\text{Enc}_{\text{sk}}(\{m_i\}_{i \in [r]}) \rightarrow \{\text{ct}[m_i]\}_{i \in [r]}$ takes the secret key sk and plaintexts $\{m_i\}_{i \in [r]} \in \mathcal{P}^r$, outputs ciphertexts $\{\text{ct}[m_i]\}_{i \in [r]} \in \mathcal{C}^r$.
- $\mathcal{E}.\text{Dec}_{\text{sk}}(\{\text{ct}[m_i]\}_{i \in [r]}) \rightarrow \{m_i\}_{i \in [r]}$ takes the secret key sk and ciphertexts $\{\text{ct}[m_i]\}_{i \in [r]} \in \mathcal{C}^r$, outputs plaintexts $\{m_i\}_{i \in [r]} \in \mathcal{P}^r$.
- $\mathcal{E}.\text{Eval}_{\text{pk}}(f, \{\text{ct}[m_i]\}_{i \in [\ell]})$ takes the public key pk , a function $f : \mathcal{P}^\ell \rightarrow \mathcal{P}^r$ and a set of ciphertexts $\{\text{ct}[m_i]\}_{i \in [\ell]} \in \mathcal{C}^\ell$, outputs ciphertexts $\{\text{ct}[m'_i]\}_{i \in [r]} \in \mathcal{C}^r$.

Correctness. An HE scheme $\mathcal{E}$ is correct for functions in $\mathcal{F}$ if for any function $f \in \mathcal{F}$ and any inputs $\{m_i\}_{i \in [\ell]}$, the relation

$$\mathcal{E}.\text{Dec}_{\text{sk}}(\mathcal{E}.\text{Eval}_{\text{pk}}(f, \text{ct}[m_1], \dots, \text{ct}[m_\ell])) = f(m_1, \dots, m_\ell),$$

holds with probability no lower than $1 - \text{negl}(\lambda)$, where $(\text{sk}, \text{pk}) \leftarrow \mathcal{E}.\text{KeyGen}(1^\lambda)$ and the ciphertexts $\{\text{ct}[m_i]\} \leftarrow \mathcal{E}.\text{Enc}_{\text{sk}}(\{m_i\})$.

CPA security. An HE scheme $\mathcal{E}$ is IND-CPA secure, if for any probabilistic polynomial-time adversary $(\mathcal{A}_1, \mathcal{A}_2)$, it holds that

$$\left| \Pr \left[b = b' \mid \begin{array}{l} b \xleftarrow{\$} \{0, 1\}; (\text{sk}, \text{pk}) \leftarrow \mathcal{E}.\text{KeyGen}(1^\lambda) \\ (m_0, m_1, \text{st}) \leftarrow \mathcal{A}_1^{\text{O}_{\text{Enc}}}(\text{pk}) \\ \text{ct}[m_b] \leftarrow \mathcal{E}.\text{Enc}_{\text{sk}}(m_b) \\ b' \leftarrow \mathcal{A}_2^{\text{O}_{\text{Enc}}}(\text{ct}[m_b], \text{pk}, \text{st}) \end{array} \right] - \frac{1}{2} \right| \leq \text{negl}(\lambda),$$

where the probability is taken over the random coins in $\mathcal{E}.\text{KeyGen}(1^\lambda)$ and $\mathcal{E}.\text{Enc}$, and O_{Enc} denotes the encryption oracle.

2.2 Verifiable computation

We define the syntax of verifiable computation [38, 66] (in the designated-verifier variant) largely following the notation of [66] in Definition 2.

Definition 2 (Verifiable computation [66]). A verifiable computation scheme $\mathcal{VC} = (\text{Setup}, \text{Compute}, \text{Verify})$ consists of the following algorithms:

- $\mathcal{VC}.\text{Setup}(1^\lambda, F) \rightarrow (\text{ek}_F, \text{vk}_F)$: On input the security parameter λ and a circuit F , the key generation algorithm outputs a public evaluation key ek_F and a private verification key vk_F .
- $\mathcal{VC}.\text{Compute}(\text{ek}_F, x) \rightarrow (y, \pi_y)$: On input the evaluation key and input x , outputs the computation result $y = F(x)$ and a proof of its correctness π_y .
- $\mathcal{VC}.\text{Verify}(\text{vk}_F, x, y, \pi_y) \rightarrow 0/1$: On input the verification key, input and output as well as the proof, returns accept (1) if $y = F(x)$ and reject (0) otherwise.

A $\mathcal{VC}$ scheme must satisfy the following two properties. A $\mathcal{VC}$ is correct (for functions in $\mathcal{F}$), if for all corresponding circuits F and $x \in \text{Domain}(F)$,

$$\Pr \left[\mathcal{VC}.\text{Verify}(\text{vk}_F, x, y, \pi_y) = 1 \mid \begin{array}{l} (\text{ek}_F, \text{vk}_F) \leftarrow \mathcal{VC}.\text{Setup}(1^\lambda, F) \\ (y, \pi_y) \leftarrow \mathcal{VC}.\text{Compute}(\text{ek}_F, x) \end{array} \right] \geq 1 - \text{negl}(\lambda).$$

A $\mathcal{VC}$ is secure/sound for a circuit F , if for any PPT adversary $\mathcal{A}$,

$$\Pr \left[\begin{array}{l} \mathcal{VC}.\text{Verify}(\text{vk}_F, \hat{x}, \hat{y}, \hat{\pi}) = 1 \\ \wedge \hat{y} \neq F(\hat{x}) \end{array} \mid \begin{array}{l} (\text{ek}_F, \text{vk}_F) \leftarrow \mathcal{VC}.\text{Setup}(1^\lambda, F) \\ (\hat{x}, \hat{y}, \hat{\pi}) \leftarrow \mathcal{A}(1^\lambda, \text{ek}_F) \end{array} \right] \leq \text{negl}(\lambda).$$

A $\mathcal{VC}$ has proof uniqueness, if for any honestly generated verification key vk_F and $y = F(x)$, there exists a unique proof π for which $\mathcal{VC}.\text{Verify}(\text{vk}_F, x, y, \pi) = 1$.

Remark 1. The third property is optional for verifiable computation and can be regarded as a non-interactive variant of the unambiguity property described in [42]. Most efficient instantiations of verifiable computation can satisfy this property when not aiming for zero-knowledge which follows from the closely related *unique response* property required for showing simulation-extractability [35, 29, 53].

2.3 Zero-knowledge proofs

Multilinear extension (MLE). Given a function $f : \{0, 1\}^\mu \rightarrow \mathbb{F}$, its *multilinear extension* is a μ -variate multilinear polynomial $\tilde{f} \in \mathbb{F}^{\leq 1}[X_1, \dots, X_\mu]$ such that $\tilde{f}(\mathbf{x}) = f(\mathbf{x})$ for all $\mathbf{x} \in \{0, 1\}^\mu$ and degree at most one in each variable. It can be computed as

$$\tilde{f}(\mathbf{x}) = \sum_{\mathbf{e} \in \{0, 1\}^\mu} f(\mathbf{e}) \cdot \tilde{\text{eq}}(\mathbf{x}, \mathbf{e}) = \sum_{\mathbf{e} \in \{0, 1\}^\mu} f(\mathbf{e}) \cdot \prod_{i=1}^{\mu} (x_i \cdot e_i + (1 - x_i) \cdot (1 - e_i)),$$

where $\tilde{\text{eq}}(\cdot, \mathbf{e})$ denotes the MLE of the equality function

$$\text{eq}(\cdot, \mathbf{e}) : \{0, 1\}^\mu \rightarrow \mathbb{F} : \mathbf{x} \mapsto \begin{cases} 1 & \text{if } \mathbf{x} = \mathbf{e} \\ 0 & \text{otherwise} \end{cases}.$$

Polynomial commitment scheme (PCS) for multilinear polynomials. Following [14, 43, 74], our work uses a *transparent* PCS that does not require a trusted setup. We recall the definition below.

Definition 3 (Polynomial commitment scheme [14, 43, 74]). A polynomial commitment scheme for multilinear polynomials over a field $\mathbb{F}$ is a tuple of four protocols $\mathcal{PC} = (\text{Setup}, \text{Commit}, \text{Open}, \text{Eval})$:

- $\text{Setup}(1^\lambda, \mu) \rightarrow \text{pp}$ takes security parameter λ and the number of variables μ in a multilinear polynomial, outputs public parameters pp .
- $\text{Commit}(\text{pp}, \mathcal{G}) \rightarrow C$ takes a multilinear polynomial $\mathcal{G} \in \mathbb{F}^{\leq 1}[X_1, \dots, X_\mu]$, outputs a commitment C .
- $\text{Open}(\text{pp}, C, \mathcal{G}) \rightarrow b$ takes a commitment C and a multilinear polynomial $\mathcal{G} \in \mathbb{F}^{\leq 1}[X_1, \dots, X_\mu]$, outputs a bit $b \in \{0, 1\}$.
- $\text{Eval}(\text{pp}, C, \mu, \mathbf{z}, y; \mathcal{G})$ is an interactive public-coin protocol between a probabilistic polynomial-time (PPT) prover P and a verifier V . Both P and V hold a commitment C , the number of variables μ , an evaluation point $\mathbf{z} \in \mathbb{F}^\mu$ and a value $y \in \mathbb{F}$. P additionally knows a μ -variate multilinear polynomial $\mathcal{G} \in \mathbb{F}^{\leq 1}[X_1, \dots, X_\mu]$. P wants to convince V that $\mathcal{G}$ is an opening to C and $\mathcal{G}(\mathbf{z}) = y$. At the end of the protocol, V outputs a bit $b \in \{0, 1\}$.

Completeness. $\mathcal{PC}$ is complete if for any $\mu \in \mathbb{N}$, any multilinear polynomial $\mathcal{G} \in \mathbb{F}^{\leq 1}[X_1, \dots, X_\mu]$ and any evaluation point $\mathbf{z} \in \mathbb{F}^\mu$, it holds that

$$\Pr \left[\text{Eval}(\text{pp}, C, \mu, \mathbf{z}, \mathcal{G}(\mathbf{z}); \mathcal{G}) = 1 \mid \begin{array}{l} \text{pp} \leftarrow \text{Setup}(1^\lambda, \mu) \\ C \leftarrow \text{Commit}(\text{pp}, \mathcal{G}) \end{array} \right] = 1.$$

Binding. $\mathcal{PC}$ is binding if for any $\mu \in \mathbb{N}$ and PPT adversary $\mathcal{A}$, it holds that

$$\Pr \left[\mathcal{G}_0 \neq \mathcal{G}_1 \mid \begin{array}{l} \text{pp} \leftarrow \text{Setup}(1^\lambda, \mu) \\ (C, \mathcal{G}_0, \mathcal{G}_1) \leftarrow \mathcal{A}(\text{pp}) \\ \text{acc} \leftarrow \text{Open}(\text{pp}, C, \mathcal{G}_0) \\ \text{acc} \leftarrow \text{Open}(\text{pp}, C, \mathcal{G}_1) \end{array} \right] \leq \text{negl}(\lambda).$$

Knowledge soundness. $\mathcal{PC}$ is knowledge sound if Eval is an argument of knowledge for the relation

$$\mathbf{R}_{\text{Eval}, \text{pp}} = \left\{ \left(\begin{array}{l} x = (C, \mathbf{z}, y) \\ w = (\mathcal{G}) \end{array} \right) \mid \mathcal{G}(\mathbf{z}) = y \wedge \text{Open}(\text{pp}, C, \mathcal{G}) = \text{acc} \right\},$$

where $\text{pp} \leftarrow \text{Setup}(1^\lambda, \mu)$.

The sum-check protocol. For any $\mathcal{G} \in \mathbb{F}^{\leq \ell}[X_1, \dots, X_\mu]$, the sum-check protocol [61] enables a verifier V to verify the claim

$$H = \sum_{x_1 \in \{0,1\}} \cdots \sum_{x_\mu \in \{0,1\}} \mathcal{G}(x_1, \dots, x_\mu) \quad (2)$$

via interaction with a prover P over μ rounds, one for each variable of $\mathcal{G}$. This is achieved by reducing the verification of Eq. (2) to the verification of a single evaluation of $\mathcal{G}$ at a random point $(r_1, \dots, r_\mu)$.

In terms of performance, the communication between P and V is $\mathcal{O}(\mu \cdot \ell)$ field elements. The verifier cost is proportional to the total communication, plus a single oracle query to $\mathcal{G}$ to obtain $\mathcal{G}(r_1, \dots, r_\mu)$. This is significantly lower than directly computing Eq. (2) by evaluating $\mathcal{G}$ at all 2^μ points.

2.4 Wrong key decryption

Following [13], we define a secret-key HE scheme with plaintext space $\mathcal{P}$ and ciphertext space $\mathcal{C}$ in Section 2.1. In addition to correctness and IND-CPA security, we require the scheme to provide a multi-message wrong-key decryption property, which extends wrong-key decryption [58, 59, 33]. Multi-message wrong-key decryption is satisfied by RLWE-based HE schemes (including GBFV [37]) under the hardness of RLWE and the circular security assumption.

Definition 4 (Multi-message wrong key decryption). *Fix integers d, n and an HE scheme $(\mathcal{E}.\text{KeyGen}, \mathcal{E}.\text{Enc}, \mathcal{E}.\text{Eval}, \mathcal{E}.\text{Dec})$ with plaintext space $\mathcal{P}$. Let $(\text{sk}_i, \text{pk}_i)$ denote a secret key and public evaluation key pair of the HE scheme, for $i \in [d]$. Then, the HE scheme has the multi-message wrong-key decryption property, if for any $(v_1, \dots, v_n) \in \mathcal{P}^n$,*

$$(\{\text{pk}_i\}_{i=1}^d, \hat{v}_1, \dots, \hat{v}_n) \stackrel{c}{\approx} (\{\text{pk}_i\}_{i=1}^d, r_1, \dots, r_n),$$

where $\hat{v}_\eta \leftarrow \text{Dec}_{\text{sk}_1}(\text{Enc}_{\text{sk}_j}(v_\eta))$, for $\eta \in [n]$ and any $j \neq 1$, and $r_\eta \leftarrow \mathcal{P}$, for $\eta \in [n]$. In words, wrong-key decryptions of any n values are computationally indistinguishable from random plaintext values.

2.5 Verifiable computation on encrypted data

Outsourcing a client's computation by sending only an encryption of x (the client's input) leads to *verifiable computation on encrypted data* (vCOED). Its syntax is defined in Definition 5 when the server does not have input. Note that our definition allows the use of the secret key sk during the input encryption routine ProbGen.

Definition 5 (Verifiable computation on encrypted data [31, 76]). *A scheme for verifiable computation on encrypted data consists of the following four algorithms:*

- $\text{Setup}(1^\lambda, C) \rightarrow (\text{sk}, \text{pk})$: On input the security parameter and a circuit C defining the computation, the setup algorithm returns a keypair (sk, pk) .
- $\text{ProbGen}(\text{sk}, x) \rightarrow (\text{Enc}(x), \text{st})$: On input the secret key from the setup sk and input to the computation x , the input encryption routine returns an encryption of x , $\text{Enc}(x)$ and a client state st .

- $\text{Compute}(\text{pk}, \text{Enc}(x), C) \rightarrow (\text{Enc}(y), \pi)$: On input the public key from Setup, an encryption of the client's input, and the circuit, the server's computation returns an encryption of the result $\text{Enc}(y)$ and a proof π .
- $\text{Verify}(\text{sk}, \text{Enc}(y), \text{st}, \pi) \rightarrow y/\perp$: On input the secret key sk , encryption of the computation result $\text{Enc}(y)$, the client's state st and the proof π , the verification algorithm returns y if indeed $y = C(x)$ and returns $\perp$ (abort) otherwise.

Correctness. A vCOED scheme is correct if the computation of an honest server always passes the verification and decryption of an honest client. This is formalized in Definition 6.

Definition 6 (Correctness). A vCOED scheme $(\text{Setup}, \text{ProbGen}, \text{Compute}, \text{Verify})$ for a circuit C is correct, if, for all inputs x , the following holds.

$$\Pr \left[\text{Verify}(\text{sk}, \text{Enc}(y), \text{st}, \pi) = C(x) \mid \begin{array}{l} (\text{sk}, \text{pk}) \leftarrow \text{Setup}(1^\lambda, C), \\ (\text{Enc}(x), \text{st}) \leftarrow \text{ProbGen}(\text{sk}, x), \\ (\text{Enc}(y), \pi) \leftarrow \text{Compute}(\text{pk}, \text{Enc}(x), C) \end{array} \right] \geq 1 - \text{negl}(\lambda).$$

For efficiency, we require *outsourcability*, i.e. a vCOED scheme is outsourceable if the running time of ProbGen and Verify as well as the size of π is $o(|C|)$, where $|C|$ denotes the size of the circuit.

Soundness. A vCOED scheme is sound if a malicious server cannot convince the client to accept an incorrect output, except with negligible probability.

Definition 7 (Soundness). A vCOED scheme $(\text{Setup}, \text{ProbGen}, \text{Compute}, \text{Verify})$ is sound for a circuit C , if for any PPT algorithm $\mathcal{A}$ and any input x , the following holds.

$$\Pr \left[\begin{array}{l} \text{Verify}(\text{sk}, \text{Enc}(\hat{y}), \text{st}, \hat{\pi}) = \hat{y} \\ \wedge \hat{y} \neq C(x) \end{array} \mid \begin{array}{l} (\text{sk}, \text{pk}) \leftarrow \text{Setup}(1^\lambda, C), \\ (\text{Enc}(x), \text{st}) \leftarrow \text{ProbGen}(\text{sk}, x), \\ (\text{Enc}(\hat{y}), \hat{\pi}) \leftarrow \mathcal{A}(\text{pk}, \text{Enc}(x), C) \end{array} \right] \leq \text{negl}(\lambda).$$

Security. We require strong client privacy⁷, also termed client privacy with a verification oracle in [76]. Intuitively, a malicious server cannot distinguish two inputs from the client even if it learns the verification result of its chosen query (once). This is formalized in the game $\mathsf{G}_{\mathcal{A}, C}^{\text{privacy}}$ in Fig. 1. Definition 8 defines the security requirement. For our applications, we only need to allow a single verification query as in [76], whereas the stronger definition of [31] allows a polynomial number.

Definition 8 (Strong client privacy [31, 76]). A vCOED scheme has strong client privacy for a circuit C , if for any tuple of PPT algorithms $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, the following holds.

$$\text{Adv}_{\mathcal{A}, C}^{\text{privacy}}(\lambda) = \left| \Pr \left[\mathsf{G}_{\mathcal{A}, C}^{\text{privacy}}(\lambda) = 1 \right] - \frac{1}{2} \right| \leq \text{negl}(\lambda).$$

⁷ This is analogous to Indistinguishability under Ciphertext Verification Attacks (IND-CVA) for symmetric encryption schemes with integrity protection [45, 47].

$G_{\mathcal{A},C}^{\text{privacy}}(\lambda)$	$O_{\text{verify}}(\text{Enc}(y), \pi)$
1 : $b \leftarrow_{\$} \{0, 1\}$	the oracle has access to sk and st in the game
2 : $(\text{sk}, \text{pk}) \leftarrow \text{Setup}(1^\lambda, C)$	and runs the instructions below at most once
3 : $(x_0, x_1, \text{st}_{\mathcal{A}}) \leftarrow \mathcal{A}_1(1^\lambda, \text{pk}, C)$	1 : $\delta_y \leftarrow \text{Verify}(\text{sk}, \text{Enc}(y), \text{st}, \pi)$
4 : $(\text{Enc}(x_b), \text{st}) \leftarrow \text{ProbGen}(\text{sk}, x_b)$	2 : return 1 iff. $\delta_y \neq \perp$ else 0
5 : $b' \leftarrow \mathcal{A}_2^{\text{O}_{\text{verify}}}(\text{st}_{\mathcal{A}}, \text{Enc}(x_b))$	
6 : return 1 iff. $b = b'$	

Fig. 1: The security game $G_{\mathcal{A},C}^{\text{privacy}}$ for strong client privacy [31, 76] against $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$.

As shown in [76], strong client privacy together with knowledge soundness allows to simulate the server's view of the interaction with the client, thereby providing client-side simulation security. This means in particular that also knowing the result of the verification does not leak any information of the client's input. Since we focus on delegation settings where server's inputs are public and incorporated into the description of the function C , soundness suffices in this setting, as summarized in Theorem 1.

Theorem 1 (Soundness and strong client privacy imply client-side simulation security [76]). *Consider a vCOED scheme $\mathcal{VE}$ for circuits $C(x)$, where x is the client's private input and the server has no private input. If $\mathcal{VE}$ satisfies soundness and strong client privacy, then it provides client-side simulation security.*

2.6 Blind proofs constructions

A blind proof scheme is a vCOED scheme with weak client privacy built upon an HE scheme $\mathcal{E}$ and a VC scheme $\mathcal{VC}$. We recall the syntax of VC in Section 2.2. A blind proof scheme [51] consists of the following algorithms:

- $\text{Setup}(1^\lambda, C) \rightarrow (\text{sk}, \text{pk})$: On input the security parameter and a circuit C , the setup algorithm runs $\mathcal{E}.\text{KeyGen}(1^\lambda) \rightarrow (\text{sk}_{\mathcal{E}}, \text{pk}_{\mathcal{E}})$ and $\mathcal{VC}.\text{Setup}(1^\lambda, C) \rightarrow (\text{ek}_C, \text{vk}_C)$, and returns $\text{sk} = (\text{sk}_{\mathcal{E}}, \text{vk}_C)$, $\text{pk} = (\text{pk}_{\mathcal{E}}, \text{ek}_C)$.
- $\text{ProbGen}(\text{sk}, x) \rightarrow (\text{Enc}(x), \text{st})$: On input the secret key $\text{sk} = (\text{sk}_{\mathcal{E}}, \text{vk}_C)$ and input to the computation x , runs $\mathcal{E}.\text{Enc}_{\text{sk}_{\mathcal{E}}}(x) \rightarrow \text{Enc}(x)$, and outputs $\text{Enc}(x)$ as well as a client state $\text{st} = x$.
- $\text{Compute}(\text{pk}, \text{Enc}(x), C) \rightarrow (\text{Enc}(y), \pi)$: On input the public key $\text{pk} = (\text{pk}_{\mathcal{E}}, \text{ek}_C)$, an encryption of the client's input, and the circuit, the server computes $\mathcal{E}.\text{Eval}_{\text{pk}}(\mathcal{VC}.\text{Compute}(\text{ek}_C, \cdot), \text{Enc}(x)) \rightarrow (\text{Enc}(y), \text{ct}[\pi_y])$, and outputs an encryption of the result $\text{Enc}(y)$ and a proof $\pi = \text{ct}[\pi_y]$.
- $\text{Verify}(\text{sk}, \text{Enc}(y), \text{st}, \pi) \rightarrow y/\perp$: On input the secret key $\text{sk} = (\text{sk}_{\mathcal{E}}, \text{vk}_C)$, encryption of the computation result $\text{Enc}(y)$, the client's state $\text{st} = x$ and the

proof $\pi = \text{ct}[\pi_y]$, the verification algorithm runs $\mathcal{E}.\text{Dec}_{\text{sk}}(\text{Enc}(y), \text{ct}[\pi_y]) \rightarrow (y, \pi_y)$ and $\mathcal{VC}.\text{Verify}(\text{vk}_C, x, y, \pi_y) \rightarrow b$. If $b = 1$, then return y ; otherwise, return $\perp$.

As a remark, Blind SNARKs [34] instantiate blind proofs but incorporate a minor modification of the above definition. Since realizing $\mathcal{VC}.\text{Compute}(\text{ek}_C, \cdot)$ with SNARKs typically involves multiple hash function computations, the homomorphic evaluation $\mathcal{E}.\text{Eval}_{\text{pk}}(\mathcal{VC}.\text{Compute}(\text{ek}_C, \cdot), \text{Enc}(x))$ is prohibitively expensive in practice. Fortunately, as shown in [34], it suffices to evaluate $\mathcal{VC}.\text{Compute}(\text{ek}_C, \cdot)$ up to the BCS compilation [8]. The challenges are derived deterministically from HE ciphertexts. As such, the output of blind proof server computation should also contain ciphertexts $\text{ct}[q]$ that encrypt responses to the simulated queries, which will be required later for the verification. We refer to [34] for the discussion on the soundness property.

3 Homomorphic encryption over random encodings

To prevent reaction-based attacks such as the one-bit forge attack [78, 76], we structure the encryption into two layers: a randomized encoding that is then encrypted using HE. If a protocol verifies the computation at the encoding level, the encoding should not leak (in-)equality of the encoded values. We formalize this concept in Section 3.1 and introduce IND-CPA-E security. In Section 3.2, we build vCOED protocols using IND-CPA-E secure HERE schemes combined with blind proofs, and prove such vCOED protocols are correct, sound, and achieve strong client privacy.

3.1 Definition and basic properties

We formalize homomorphic encryption over random encodings (HERE) in the symmetric-key variant in Definition 9.

Definition 9 (Homomorphic encryption over random encodings (HERE)).

A HERE scheme $\mathcal{ER}$ is defined by combining a random encoding $\mathcal{R}$, whose message space is $\mathcal{P}$ and encoding space is $\mathcal{P}_e$, and a homomorphic encryption scheme $\mathcal{E}$, whose plaintext space is $\mathcal{P}_e$ and ciphertext space is $\mathcal{C}$. A HERE scheme $\mathcal{ER} = (\text{KeyGen}, \text{Encode}, \text{Encrypt}, \text{Eval}, \text{Decrypt}, \text{Decode})$ has the following syntax:

- $\text{KeyGen}(1^\lambda, \mathcal{F}) \rightarrow (\text{sk}, \text{pk})$: On input the security parameter and a function family $\mathcal{F}$, outputs a secret key $\text{sk} = (\text{sk}_1, \text{sk}_2)$ and a public key pk .
- $\text{Encode}(\text{sk}_1, x) \rightarrow c_x$: On input the encoding secret key sk_1 and a value $x \in \mathcal{P}$, the probabilistic algorithm outputs an encoding $c_x \in \mathcal{P}_e$.
- $\text{Encrypt}(\text{sk}_2, c_x) \rightarrow \text{ct}_x$: On input the secret key sk_2 and an encoding $c_x \in \mathcal{P}_e$, the probabilistic algorithm outputs a ciphertext $\text{ct}_x \in \mathcal{C}$.
- $\text{Eval}_f(\text{pk}, \text{ct}_{x_1}, \dots, \text{ct}_{x_n}) \rightarrow \text{ct}_y$: On input the public key pk , n ciphertexts $\text{ct}_{x_1}, \dots, \text{ct}_{x_n} \in \mathcal{C}$, and parametrized by the function $f \in \mathcal{F}$, which takes n inputs, outputs the ciphertext of the function result $\text{ct}_y \in \mathcal{C}$.

- $\text{Decrypt}(\text{sk}_2, \text{ct}_y) \rightarrow c_y$: On input the encryption secret key sk_2 and a ciphertext $\text{ct}_y \in \mathcal{C}$, outputs an encoding $c_y \in \mathcal{P}_e$.
- $\text{Decode}(\text{sk}_1, c_y) \rightarrow y$: On input the encoding secret key sk_1 and an encoding $c_y \in \mathcal{P}_e$, outputs the underlying value $y \in \mathcal{P}$.

We define the two operations $\text{EnTEnc}(\text{sk}, x)$ and $\text{DecTDe}(\text{sk}, \text{ct})$ as encoding followed by encryption, and decryption followed by decoding, respectively. Formally,

$$\begin{aligned}\text{EnTEnc}(\text{sk}, x) &:= \text{Encrypt}(\text{sk}_2, \text{Encode}(\text{sk}_1, x)), \\ \text{DecTDe}(\text{sk}, \text{ct}) &:= \text{Decode}(\text{sk}_1, \text{Decrypt}(\text{sk}_2, \text{ct})).\end{aligned}$$

Looking ahead, a HERE scheme outputs HERE ciphertexts, which, depending on the construction, may consist of multiple HE ciphertexts. In this section, when writing ciphertext we implicitly mean HERE ciphertexts.

A HERE scheme is correct, if it allows homomorphic operations on the ciphertexts. Formally, it should satisfy the two properties in Definition 10.

Definition 10 (Encoding (Evaluation) Correctness). *We say a homomorphic encryption scheme over random encodings $\mathcal{ER} = (\text{KeyGen}, \text{Encode}, \text{Encrypt}, \text{Eval}, \text{Decrypt}, \text{Decode})$ has correct encoding evaluation w.r.t. the function $f \in \mathcal{F}$, if there is a deterministic function $F : \mathcal{P}_e^n \rightarrow \mathcal{P}_e$, s.t. for all $\{x_i\} \in \mathcal{P}^n$,*

$$\Pr \left[\text{Decrypt}(\text{sk}_2, \text{ct}_y) = F(\{c_{x_i}\}) \mid \begin{array}{l} ((\text{sk}_1, \text{sk}_2), \text{pk}) \leftarrow \text{KeyGen}(1^\lambda, \mathcal{F}), \\ \{c_{x_i}\} \leftarrow \text{Encode}(\text{sk}_1, \{x_i\}), \\ \{\text{ct}_{x_i}\} \leftarrow \text{Encrypt}(\text{sk}_2, \{c_{x_i}\}), \\ \text{ct}_y \leftarrow \text{Eval}_f(\text{pk}, \{\text{ct}_{x_i}\}) \end{array} \right] \geq 1 - \text{negl}(\lambda),$$

where the probability is taken over the random coins in KeyGen , Encode and Encrypt . Further, we say its encoding is correct w.r.t. the function $f \in \mathcal{F}$, if

$$\Pr \left[\text{Decode}(\text{sk}_1, F(\{c_{x_i}\})) = f(\{x_i\}) \mid \begin{array}{l} ((\text{sk}_1, \text{sk}_2), \text{pk}) \leftarrow \text{KeyGen}(1^\lambda, \mathcal{F}), \\ \{c_{x_i}\} \leftarrow \text{Encode}(\text{sk}_1, \{x_i\}) \end{array} \right] \geq 1 - \text{negl}(\lambda),$$

where the probability is taken over the random coins in KeyGen and Encode .

The two properties, correct encoding evaluation and encoding correctness imply the expected correctness property: decrypting and decoding an honestly evaluated HERE ciphertext yields the expected function output.

To define security, we extend the classic notion of indistinguishability against chosen plaintext attacks (IND-CPA) by adding an equality test oracle, and name this notion IND-CPA-E. The equality oracle tells the IND-CPA-E adversary if two queried ciphertexts ct_c and ct_u decrypt to the same *encoding* (not to the same encoded value!), i.e. the oracle returns 1 if $\text{Decrypt}(\text{sk}_2, \text{ct}_c) = \text{Decrypt}(\text{sk}_2, \text{ct}_u)$ over a suitably defined equality relation over the encoding space $\mathcal{P}_e$. The oracle can only be called once with q ciphertext pairs and replies with 1 if all q pairs are pairwise equal. Intuitively, this models the randomized nature of the encoding that is encrypted but restricts the adversary to non-adaptive queries. The formal definition is given in Definition 11.

$G_{\mathcal{A}}^{\text{ind-e}}(\lambda)$	$O_{\text{En}}(x^{(i)})$
1 : $b \leftarrow_{\$} \{0, 1\}$	1 : return $\text{EnTEnc}(\text{sk}, x^{(i)})$
2 : $((\text{sk}_1, \text{sk}_2), \text{pk}) \leftarrow \text{KeyGen}(1^\lambda, \mathcal{F})$	$O_{\text{Equal}}\left((\text{ct}_u^{(j)}, \text{ct}_v^{(j)})_{j=1}^q\right)$
3 : $(\vec{x}_0, \vec{x}_1, \text{st}_{\mathcal{A}}) \leftarrow \mathcal{A}_1^{\text{OEn}}(1^\lambda, \text{pk})$	this oracle runs at most once
4 : $\vec{\text{ct}}^* \leftarrow \text{EnTEnc}(\text{sk}, \vec{x}_b)$	1 : $u^{(j)} \leftarrow \text{Decrypt}(\text{sk}_2, \text{ct}_u^{(j)})$, for $j \in [q]$
5 : $b' \leftarrow \mathcal{A}_2^{\text{OEn}, \text{OEqual}}(1^\lambda, \text{st}_{\mathcal{A}}, \vec{\text{ct}}^*)$	2 : $v^{(j)} \leftarrow \text{Decrypt}(\text{sk}_2, \text{ct}_v^{(j)})$, for $j \in [q]$
6 : return 1 iff. $b = b'$	3 : return 1 iff. $\bigwedge_{j=1}^q u^{(j)} = v^{(j)}$

Fig. 2: The IND-CPA-E game $G_{\mathcal{A}}^{\text{ind-e}}$ (indistinguishability under chosen plaintext attack with equality test oracle). The game is valid if $|\vec{x}_0| = |\vec{x}_1|$ and if all oracle inputs are in the correct domains.

Definition 11 (IND-CPA-E security). We say a homomorphic encryption scheme over random encodings (KeyGen , Encode , Encrypt , Eval , Decrypt , Decode) for functions in $\mathcal{F}$ is IND-CPA-E secure, if for all PPT adversaries $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, the advantage in the game $G_{\mathcal{A}}^{\text{ind-e}}$ (see Fig. 2) is

$$\text{Adv}_{\mathcal{A}}^{\text{ind-e}}(\lambda) = \left| \Pr[G_{\mathcal{A}}^{\text{ind-e}}(\lambda) = 1] - \frac{1}{2} \right| \leq \text{negl}(\lambda).$$

3.2 vCOED with strong client privacy from HERE and blind proofs

Next, we show that a HERE scheme achieves strong client privacy for vCOED. We apply the blind proof [38, 36, 2, 34, 76] paradigm to verify correct computation on the encodings. While the blind proof now needs to verify the homomorphic operations of the encoding instead of the computation alone, we believe that this verification is still *simple*. A secure encoding only needs to *randomize* the computation instead of *hiding underlying values*. Providing indistinguishability of ciphertexts is passed to the homomorphic encryption scheme instead which does not need to be checked by the blind proof. Since such a randomized computation can be realized on the plaintext domain and with less complex operations, the verifiable computation scheme can be utilized in favourable parameter regimes.

Our vCOED construction. We detail our construction in Fig. 3. We combine a secure homomorphic encryption scheme over random encodings with a verifiable computation scheme $\mathcal{VC}$ in the blind proof paradigm. The $\mathcal{VC}$ scheme will verify the operations on encoded values. Since our encoding is simple, these operations are much more efficient to verify than operations on the ciphertext.

The properties of correctness and soundness follow naturally from blind proofs, assuming the HE scheme has sufficient evaluation capacity. Below we

vCOED scheme with strong client privacy from HERE

Parameters: Let $\mathcal{ER} = (\text{KeyGen}, \text{Encode}, \text{Encrypt}, \text{Eval}, \text{Decrypt}, \text{Decode})$ be a homomorphic encryption scheme over random encodings for a circuit C . Let F be the circuit to evaluate C homomorphically over the encoding space $\mathcal{P}_e$. Let $\mathcal{VC} = (\mathcal{VC}.\text{Setup}, \mathcal{VC}.\text{Compute}, \mathcal{VC}.\text{Verify})$ be a verifiable computation scheme for F .

Setup($1^\lambda, C$): Output $\text{sk} = (\text{sk}_1, \text{sk}_2, \text{vk})$ and $\text{pk} = (\text{pk}_{\mathcal{ER}}, \text{ek})$ obtained as

1. $(\text{sk}_1, \text{sk}_2), \text{pk}_{\mathcal{ER}} \leftarrow \mathcal{ER}.\text{KeyGen}(1^\lambda, C)$
2. $\text{ek}, \text{vk} \leftarrow \mathcal{VC}.\text{Setup}(1^\lambda, F)$

ProbGen($\text{sk}, \vec{x}$): Parse $\text{sk} = (\text{sk}_1, \text{sk}_2, \text{vk})$ and let $\vec{x} \in \mathcal{P}^n$.

1. Compute $\vec{c} \leftarrow \mathcal{ER}.\text{Encode}(\text{sk}_1, \vec{x})$. $(\vec{c} \in \mathcal{P}_e^n)$
2. Compute $\vec{\text{ct}} \leftarrow \mathcal{ER}.\text{Encrypt}(\text{sk}_2, \vec{c})$. $(\vec{\text{ct}} \in \mathcal{C}^n)$
3. Set $\text{st} = (\vec{c})$.

Output $\text{Enc}(\vec{x}) = \vec{\text{ct}}$ and client state st .

Compute($\text{pk}, \text{Enc}(\vec{x}), C$): Parse pk as $(\text{pk}_{\mathcal{ER}}, \text{ek})$ and $\text{Enc}(\vec{x})$ as $\vec{\text{ct}}$. The server runs

1. $\text{ct}', \text{ct}_\pi := \mathcal{ER}.\text{Eval}(\text{pk}_{\mathcal{ER}}, \mathcal{VC}.\text{Compute}_{\text{ek}, F}, \vec{\text{ct}})$.
- Output $(\text{ct}', \text{ct}_\pi)$.

Verify($\text{sk}, \text{ct}', \text{st}, \text{ct}_\pi$): Parse st as $\vec{c}$. The client

1. decrypts the result $c_y := \mathcal{ER}.\text{Decrypt}(\text{sk}_2, \text{ct}')$. $(c_y \in \mathcal{P}_e)$
2. decrypts the proof $\pi_{c_y} := \mathcal{ER}.\text{Decrypt}(\text{sk}_2, \text{ct}_\pi)$.
3. verifies the proof $\delta \leftarrow \mathcal{VC}.\text{Verify}(\text{vk}, \vec{c}, c_y, \pi_{c_y})$
4. if $\delta = \text{acc}$, set $y = \mathcal{ER}.\text{Decode}(\text{sk}_1, c_y)$, else set $y = \perp$.

Output y .

Fig. 3: Our vCOED scheme with strong client privacy from black-box HE(RE) and VC.

focus on strong client privacy. Together, these properties ensure the client is simulatable for the server, i.e. no message bit leakage occurs even if the server learns the result of the client's verification.

Security. The following theorem establishes that our construction has strong client privacy.

Theorem 2. *Let $\mathcal{ER}$ be an IND-CPA-E secure homomorphic encryption scheme over random encodings for a circuit C and $\mathcal{VC}$ a secure verifiable computation scheme for a circuit F , where F describes the operations of C on the encodings. If $\mathcal{VC}$ supports proof uniqueness and $\mathcal{ER}$ satisfies encoding correctness and encoding evaluation correctness (see Definition 10), then the construction in Fig. 3 is a secure vCOED scheme for C with strong client privacy.*

Proof. Since correctness follows from the correctness of all components, we will now prove that our vCOED construction has strong client privacy as in Definition 8.

We construct a reduction adversary $\mathcal{B}$ for the IND-CPA-E game (cf. Definition 11) which internally runs the adversary $\mathcal{A}$ for $\mathsf{G}_{\mathcal{A},C}^{\text{privacy}}$. The reduction $\mathcal{B}$ proceeds as follows:

1. It receives the $\mathcal{ER}$ scheme public key pk from the IND-CPA-E challenger and creates a new $\mathcal{VC}$ scheme keypair, $(\text{ek}, \text{vk}) \leftarrow \mathcal{VC}.\text{Setup}(1^\lambda, F)$.
2. It invokes $\mathcal{A}$ on input pk, ek, C and receives two inputs of equal length $\vec{x}_0, \vec{x}_1$. The reduction passes $\vec{x}_0, \vec{x}_1$ to the IND-CPA-E challenger.
3. Upon receiving the challenge ciphertext $\vec{\text{ct}}$, $\mathcal{B}$ invokes $\mathcal{A}$ with the same challenge ciphertext $\vec{\text{ct}}$. The reduction provides $\mathcal{A}$ access to the verification oracle $\text{O}_{\text{verify}}(\text{ct}_y, \text{ct}_\pi)$ which is simulated by
 - (a) The reduction computes the expected Compute procedure as $\text{ct}^*, \text{ct}_\pi^* \leftarrow \mathcal{ER}.\text{Eval}(\text{pk}, \mathcal{VC}.\text{Compute}_{\text{ek}, F}, \vec{\text{ct}})$.
 - (b) The reduction invokes the equality oracle O_{Equal} on input $((\text{ct}_y, \text{ct}^*), (\text{ct}_\pi, \text{ct}_\pi^*))$, i.e. testing equality between the input to the verification oracle provided by $\mathcal{A}$ and the input produced by an honest computation.
 - (c) The simulated oracle O_{verify} returns whatever the equality oracle O_{Equal} returned in the previous step.
4. Finally, the reduction obtains $\mathcal{A}$'s guess b' and outputs b' as guess to the IND-CPA-E challenger.

The inputs $(\text{pk}, \text{ek}, C, \vec{\text{ct}})$ that $\mathcal{A}$ receives in the reduction are identical to those in the game $\mathsf{G}_{\mathcal{A},C}^{\text{privacy}}$ by construction. What remains to show is that the response from the simulated oracle is indistinguishable from the verification oracle in the protocol.

(O_{verify} in $\mathsf{G}_{\mathcal{A},C}^{\text{privacy}}$)	(O_{verify} provided by $\mathcal{B}$ via O_{Equal})
$c_y := \mathcal{ER}.\text{Decrypt}(\text{sk}_2, \text{ct}_y)$	$c_y := \mathcal{ER}.\text{Decrypt}(\text{sk}_2, \text{ct}_y)$
$\pi_{c_y} := \mathcal{ER}.\text{Decrypt}(\text{sk}_2, \text{ct}_\pi)$	$\pi_{c_y} := \mathcal{ER}.\text{Decrypt}(\text{sk}_2, \text{ct}_\pi)$
	$c_y^* := \mathcal{ER}.\text{Decrypt}(\text{sk}_2, \text{ct}^*)$
	$\pi_{c_y}^* := \mathcal{ER}.\text{Decrypt}(\text{sk}_2, \text{ct}_\pi^*)$
return $\mathcal{VC}.\text{Verify}(\text{vk}, \vec{\text{c}}, c_y, \pi_{c_y})$	return $c_y = c_y^* \wedge \pi_{c_y} = \pi_{c_y}^*$

Let δ_p denote the output in $\mathsf{G}_{\mathcal{A},C}^{\text{privacy}}$, and δ_s denote the output in $\mathcal{B}$'s simulation. We show that $\delta_p = \delta_s$ must hold by correctness, soundness, proof uniqueness of the $\mathcal{VC}$ scheme.

Case $\delta_p = 0$ and $\delta_s = 1$. Since $\delta_s = 1$, and given the encoding evaluation correctness of $\mathcal{ER}$, it holds that $c_y = c_y^* = F(\vec{\text{c}})$, and further that $\pi_{c_y} = \pi_{c_y}^* = \mathcal{VC}.\text{Compute}(\text{ek}_F, \vec{\text{c}})$. Due to the correctness of the $\mathcal{VC}$ scheme, this happens with negligible probability.

Case $\delta_p = 1$ and $\delta_s = 0$. Since $\delta_p = 1$, it must hold that $c_y = F(\vec{\text{c}})$ with high probability due to soundness of $\mathcal{VC}$. Furthermore, $c_y^* = c_y$ due to encoding

evaluation correctness of the $\mathcal{ER}$ scheme. Since $\delta_s = 0$, it must hold that $\pi_{c_y} \neq \pi_{c_y}^*$, but this contradicts the proof uniqueness property because $\delta_p = 1$.

Thus, $\text{Adv}_{\mathcal{B}}^{\text{ind-e}}(\lambda) \geq \text{Adv}_{\mathcal{A},C}^{\text{privacy}}(\lambda) - \text{negl}(\lambda)$, which concludes the proof. $\square$

Remark 2. Efficient instantiations of blind proofs will not homomorphically compute the Fiat-Shamir part of BCS compilation (see [34] for more details) and instead generate challenges by hashing ciphertext values. Theorem 2 still holds, although we cannot rely on proof uniqueness in the final part of the proof above because multiple ciphertext representations can be generated to construct different challenges. Thus we require an extra hybrid game where the VC verification in the verification oracle no longer checks the Fiat-Shamir generation of these challenge values. This cannot improve the advantage of the adversary because it is a part of VC verification that they could compute without access to sk . Lastly, to compute step 3a, adversary $\mathcal{B}$ uses the same challenge values that can be derived from ct_{π} .

4 Instantiating HERE

A *heavyweight* construction of HERE uses HE as the randomised encoding, resulting in a two-layer HE structure: the inner layer is an HE scheme with secret key sk_{inn} , and the outer layer is another HE scheme with secret key sk_{out} whose plaintext space is the ciphertext space of the inner HE scheme. The evaluation phase of HERE then amounts to homomorphically evaluating the function $\text{Eval}(f, \text{Enc}_{\text{sk}_{\text{inn}}}(m))$ under the outer scheme. In other words, this construction provides a specialized vHE protocol that proves relations over ciphertexts of the inner scheme. Client reactions can be exploited to infer information about the inner ciphertexts, which do not leak underlying values due to the IND-CPA security. This idea has been applied in [68] to verifiably compute PIR hints, where the inner HE is LWE-based and the outer HE is RLWE-based encryption with a 32-bit plaintext modulus, which matches the LWE ciphertext modulus.

On the other hand, a secure HERE encoding only needs to randomize the computation, rather than hide underlying values. Therefore, in this section we instantiate HERE with a more *lightweight* encoding. Its IND-CPA-E security relies on a bounded-depth circuit-awareness assumption: the adversary can only submit well-formed ciphertexts (i.e. outputs of the encryption and evaluation oracles) to the equality oracle.

4.1 The Encoding

We build a homomorphic encryption scheme over random encodings for degree- d multivariate polynomials from a levelled homomorphic encryption scheme with plaintext domain $\mathbb{F}_p$ and encoding domain $\mathbb{F}_p^{d+1}$. Let $\mathcal{E} = (\mathcal{E}.\text{KeyGen}, \mathcal{E}.\text{Enc}, \mathcal{E}.\text{Eval}, \mathcal{E}.\text{Dec})$ denote the operations of the HE scheme.

During key generation, $d+1$ distinct secret keys together with the corresponding public evaluation keys of the HE scheme are generated. We also fix $d+1$

distinct, non-zero evaluation points $\alpha_1, \dots, \alpha_{d+1} \in \mathbb{F}_p$, which can be public, e.g. $\alpha_i = i$. An input vector $\vec{x} \in \mathbb{F}_p^n$ is then encoded by element-wise secret-sharing $\vec{x}$ on a degree-1 polynomial following Shamir secret sharing. Concretely, we sample n degree-1 polynomials $\vec{f}$ at random s.t. $\vec{f}(0) = \vec{x}$. Then, the i -th component of the encoding is $\vec{c}_i := \vec{f}(\alpha_i)$, for $i \in [d+1]$. Finally, each component $\vec{c}_i$ is encrypted using the i -th HE key pair. We detail the construction in Fig. 4. With slight abuse of notation, we write $\text{ct}_{x,i} = \text{ct}[c_{x,i}]$, where $c_{x,i}$ is the i -th encoding component.

Evaluation. Since $d+1$ Shamir secret shares, i.e. point evaluations of the same degree-1 polynomial, are linearly homomorphic and up to d times multiplicatively homomorphic, this encoding supports the evaluation of degree- d polynomials.

Let $\text{ct}_{x,1}, \dots, \text{ct}_{x,d+1}$ and $\text{ct}_{y,1}, \dots, \text{ct}_{y,d+1}$ denote encrypted encodings of x and y under the same evaluation points $\alpha_1, \dots, \alpha_{d+1}$, respectively. Linear and non-linear operations are computed separately on each component of the encrypted encoding. Let $P(X, Y)$ be the degree- d multivariate polynomial that should be evaluated, then the output ciphertexts $\hat{\text{ct}}_1, \dots, \hat{\text{ct}}_{d+1}$ are computed as

$$\hat{\text{ct}}_i := \mathcal{E}.\text{Eval}_P(\text{pk}_i, \text{ct}_{x,i}, \text{ct}_{y,i}), \quad \text{for } i \in [d+1].$$

Decoding. Let $\hat{\text{ct}}_1, \dots, \hat{\text{ct}}_{d+1}$ be an output ciphertext returned by the server. The client decrypts each $\hat{\text{ct}}_i$ to $\hat{c}_i$, before interpolating the polynomial defined by points $(\alpha_i, \hat{c}_i)$ at 0 to obtain the result. This interpolation can be done with, e.g. Lagrange interpolation. The Lagrange coefficient L_i only depends on the fixed, public α values. Thus, decoding amounts to computing $d+1$ field multiplications and d field additions: $z = \sum_{i=1}^{d+1} L_i \cdot \hat{c}_i$.

Since decoding can be performed without knowledge of the encoding randomness $\vec{f}$, our instantiation does not have an encoding secret key sk_1 .

4.2 Bounded-depth circuit awareness

Before proceeding to prove security, we detail our assumption on the adversarial behaviour of the server. This is formalized by the notion of a circuit extractor in Definition 12.

Intuition and context. Intuitively, the definition models that if the adversary produces a ciphertext, it knows an equivalent computation it could perform on the plaintext instead. Consider a game where the adversary has access to the HE public evaluation key and to an oracle that returns ciphertexts of values the adversary does not know. Then, for any ciphertext $\text{ct}_{\mathcal{A}}$ the adversary outputs, $\mathcal{A}$ also knows a circuit F (with built-in, chosen constants of bounded depth L) that computes the plaintext hidden in $\text{ct}_{\mathcal{A}}$, when given plaintext values of the oracle responses as input. Our definition therefore captures an algebraic adversary which outputs ciphertexts computed from arithmetic circuits of L -bounded depth.

The circuit extractor definition. We give the full, formal description in Definition 12.

HERE for degree-bounded functions

Parameters: Finite field $\mathbb{F}_p$ with p prime, a (levelled) HE scheme $(\mathcal{E}.\text{KeyGen}, \mathcal{E}.\text{Enc}, \mathcal{E}.\text{Eval}, \mathcal{E}.\text{Dec})$ over plaintext $\mathbb{F}_p$. Let $\mathcal{F}$ denote the function family of multivariate degree- d polynomials, and $\alpha_i = i$, for $i \in [d+1]$.

KeyGen $(1^\lambda, \mathcal{F})$: For $i \in [d+1]$, run $(\text{sk}_{\mathcal{E},i}, \text{pk}_{\mathcal{E},i}) \leftarrow \mathcal{E}.\text{KeyGen}(1^\lambda)$.

Set $\text{sk}_1 = \emptyset$ and $\text{sk}_2 = (\text{sk}_{\mathcal{E},i})_{i=1}^{d+1}$, and output $\text{sk} = (\text{sk}_1, \text{sk}_2)$ and $\text{pk} = (\text{pk}_{\mathcal{E},i})_{i=1}^{d+1}$.

Encode $(\text{sk}_1, \vec{x})$: Let $\vec{x} \in \mathbb{F}_p^n$, then

1. Sample $\vec{a} \leftarrow \$\mathbb{F}_p^n$
2. Compute $\vec{c}_{x,i} := \vec{x} + \alpha_i \cdot \vec{a}$, for $i \in [d+1]$.

Output $\vec{c}_{x,1}, \dots, \vec{c}_{x,d+1}$.

Encrypt $(\text{sk}_2 = (\text{sk}_{\mathcal{E},i})_{i=1}^{d+1}, (\vec{c}_{x,1}, \dots, \vec{c}_{x,d+1}))$: For $i \in [d+1]$,

1. encrypt $\vec{\text{ct}}_{x,i} \leftarrow \mathcal{E}.\text{Enc}_{\text{sk}_{\mathcal{E},i}}(\vec{c}_{x,i})$ (each element in $\vec{c}_{x,i}$ is encrypted separately and stored in the ciphertext vector)

Output $\vec{\text{ct}}_{x,1}, \dots, \vec{\text{ct}}_{x,d+1}$.

Eval $_{f \in \mathcal{F}}(\text{pk} = (\text{pk}_{\mathcal{E},i})_{i=1}^{d+1}, \vec{\text{ct}}_{x,1}, \dots, \vec{\text{ct}}_{x,d+1})$: For each $i \in [d+1]$,

1. Compute $\text{ct}_{z,i} := \mathcal{E}.\text{Eval}_{\text{pk}_{\mathcal{E},i}}(f, \vec{\text{ct}}_{x,i})$. ($\text{ct}_{z,i} := f(\vec{\text{ct}}_{x,i})$)

Output $(\text{ct}_{z,1}, \dots, \text{ct}_{z,d+1})$.

Decrypt $(\text{sk}_2, (\text{ct}_{z,1}, \dots, \text{ct}_{z,d+1}))$: For $i \in [d+1]$, $c_{z,i} := \mathcal{E}.\text{Dec}_{\text{sk}_{\mathcal{E},i}}(\text{ct}_{z,i})$.

Output $(c_{z,1}, \dots, c_{z,d+1})$.

Decode $(\text{sk}_1, (c_{z,1}, \dots, c_{z,d+1}))$: Let L_i denote the Lagrange interpolator of point α_i , interpolating at 0.

Compute and output $z := L_1 \cdot c_{z,1} + \dots + L_{d+1} \cdot c_{z,d+1}$.

Fig. 4: Our instantiation of a HERE for polynomials.

Definition 12 (Bounded-depth circuit awareness). Let $\mathcal{E}$ be a homomorphic encryption scheme, and $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, $\mathcal{A}^*$ and $\mathcal{D}$ be PPT algorithms. We say $\mathcal{A}$ is L -bounded circuit-aware, if there exists a circuit extractor $\mathcal{A}^*$, s.t. for any distinguisher $\mathcal{D}$, and any random coins $r, r^* \leftarrow \$\{0,1\}^*$, the difference of probabilities $\Pr[\text{real}]$ and $\Pr[\text{extracted}]$ defined as

$$\Pr[\text{real}] = \Pr \left[\mathcal{D}(\mathcal{A}_2(\text{st}, x)) = 1 \mid \begin{array}{l} \text{sk}, \text{pk} \leftarrow \mathcal{E}.\text{KeyGen}(1^\lambda) \\ \text{ct}[x], \text{st} \leftarrow \mathcal{A}_1^0(1^\lambda, \text{pk}; r) \\ x \leftarrow \mathcal{E}.\text{Dec}_{\text{sk}}(\text{ct}[x]) \end{array} \right],$$

and

$$\Pr[\text{extracted}] = \Pr \left[\mathcal{D}(\mathcal{A}_2(\text{st}, x)) = 1 \mid \begin{array}{l} \text{sk}, \text{pk} \leftarrow \mathcal{E}.\text{KeyGen}(1^\lambda) \\ \text{ct}[x], \text{st} \leftarrow \mathcal{A}_1^0(1^\lambda, \text{pk}; r) \\ F \leftarrow \mathcal{A}^*(1^\lambda, \text{pk}, r; r^*) \\ x_j^0 \leftarrow \mathcal{E}.\text{Dec}_{\text{sk}}(\text{ct}[x_j^0]), \text{ for } j \in [q] \\ x \leftarrow F(x_1^0, \dots, x_q^0) \end{array} \right],$$

is negligible, i.e. $\text{Adv}_D^{\text{ext}}(\lambda) = |\Pr[\text{real}] - \Pr[\text{extracted}]| \leq \text{negl}(\lambda)$, and F is a circuit of depth at most L . Above, $\mathcal{O}(x)$ denotes an arbitrary oracle. If the j -th oracle call outputs a ciphertext to $\mathcal{A}_1$, we denote this ciphertext as $\text{ct}[x_j^{\mathcal{O}}]$.

On a technical level, the circuit extractor $\mathcal{A}^*$ receives $\mathcal{A}$'s random coins r as additional input to extract F . This allows $\mathcal{A}^*$ to run the now deterministic $\mathcal{A}$ and inspect its internal state.

Remark 3. Bounded-depth circuit awareness is a strong assumption on the adversary behaviour. To the best of our knowledge, producing ciphertexts not from homomorphic evaluation has only been exploited in the context of key recovery attacks [78, 19]. Such a key-recovery attack also arises in the interactive linear-only assumption in [48]. As explained in [48, Remark 3.22], recovering the secret key requires a malicious server to submit a *super-constant* number of "bad" proofs to the verifier, thereby allowing the client to detect the attack.

We believe the impact of such attacks in our setting is limited, as our security model only considers one-time verification. Therefore, such a key recovery attack leaks at most one bit of the secret key (the concrete security loss this incurs can be measured with tools such as the Leaky Estimator [27, 28], as suggested in [2]). By contrast, our work addresses the one-bit leakage of the client's inputs/intermediate values/outputs, which can directly disclose sensitive information in HE applications such as PIR and PDTE.

Our model does not address the HE key reusability across multiple outsourcing tasks. The safest option is to generate fresh key pairs for each task. When the setup cost makes this impractical (e.g. when evaluation keys are large), the client may instead follow the approach discussed in [48, Remark 3.22] to reuse HE keys while monitoring for malicious behaviour. Once such behaviour is detected, the client must abort and regenerate HE keys.

Multi-key variant. Since our construction uses multiple HE key pairs, the adversary may mix ciphertexts encrypted under different keys. This is captured in the multi-key circuit extractor variant in Definition 13, which we defer to Appendix A.

4.3 Security

We prove the construction in Fig. 4 is IND-CPA-E secure when paired with an IND-CPA HE scheme with additional properties. The exact statement is given in Theorem 3 below.

Theorem 3. *Let $\mathcal{E}$ be an IND-CPA secure levelled homomorphic encryption scheme with plaintext space $\mathbb{F}_p$, for prime p , that*

- *supports L consecutive ciphertext-ciphertext multiplications, and*
- *additionally satisfies multi-message wrong-key decryption (Definition 4), and*
- *the adversary is multi-key, L -bounded circuit-aware (Definition 13).*

Then, the HERE scheme for bounded-degree functions of degree d in Fig. 4 using $\mathcal{E}$ is IND-CPA-E secure with

$$\text{Adv}_{\mathcal{A}}^{\text{ind-e}}(\lambda) \leq (d+1)\text{Adv}_{\mathcal{B}}^{\text{ind}}(\lambda) + 2q(d+1)\text{Adv}_{\mathcal{D}}^{\text{mk-ext}}(\lambda) + \frac{2^L}{p} + \text{negl}(\lambda),$$

where $\mathcal{A}, \mathcal{B}$ and $\mathcal{D}$ are adversaries against IND-CPA-E, IND-CPA and circuit extraction respectively, and q denotes the size of $\mathcal{A}$'s equality oracle call as in Fig. 2.

Below, we sketch the proof. The full proof can be found in Appendix B.1

Proof (Sketch). We start with the IND-CPA-E game and the protocol with $d+1$ components. Then, the adversary's query to the equality oracle $\mathcal{O}_{\text{Equal}}$ contains q pairs of encodings, each with $d+1$ components: $\text{ct}_{u,1}^{(j)}, \text{ct}_{v,1}^{(j)}, \dots, \text{ct}_{u,d+1}^{(j)}, \text{ct}_{v,d+1}^{(j)}$, for $j \in [q]$.

We modify the equality oracle $\mathcal{O}_{\text{Equal}}$ in the IND-CPA-E game first. Consider the i -th component and the j -th pair. Instead of decrypting $\text{ct}_{u,i}^{(j)}$, we use the (multi-key) circuit extractor from Definition 13 to extract the relevant circuit F , and similarly, instead of decrypting $\text{ct}_{v,i}^{(j)}$, we extract circuit G . With the circuits, we compute the expected plaintexts $u_i^{(j)}$ and $v_i^{(j)}$. We repeat this replacement until all $2q(d+1)$ circuits are extracted and none of the ciphertexts given to $\mathcal{O}_{\text{Equal}}$ are decrypted any more.

With the multi-message wrong-key decryption property from Definition 4, we can argue that all those inputs to F and G representing oracle ciphertexts encrypted under a different key that the adversary may choose to use, can be replaced with random values in the plaintext domain. As a result, the equality oracle no longer decrypts any ciphertexts: a correct-key decryption can be replaced by the underlying message, and wrong-key decryptions are substituted with random values.

Then, instead of the real challenge encoding, we give the adversary encryptions of 0. However, the equality oracle will still use the real challenge encoding in its reply. Swapping challenge ciphertexts is indistinguishable by IND-CPA security of the encryption scheme since the HE secret key is no longer needed in $\mathcal{O}_{\text{Equal}}$.

At this stage, the adversary no longer receives any values derived from the secret bit b when generating the query to $\mathcal{O}_{\text{Equal}}$. We now view the response from the equality oracle as

$$\bigwedge_{i=1}^{d+1} \bigwedge_{j=1}^q H_i^{(j)}(\vec{r}) = 0,$$

where $H_i^{(j)} = F_i^{(j)} - G_i^{(j)}$, and $\vec{r}$ denotes the inputs to F and G . Since $F_i^{(j)}(\vec{r}) = u_i^{(j)}$ and $G_i^{(j)}(\vec{r}) = v_i^{(j)}$, this is equivalent.

Wlog. the input vector $\vec{r}$ now only consists of the challenge encoding $\vec{c}_i$ and random values from wrong-key decryptions, since correct-key decryptions of encryption oracle queries can be incorporated into F and G as constants.

Finally, the challenge encoding $\vec{c}_i = \vec{x}_b + \alpha_i \cdot \vec{a}$, where $\vec{x}_b$ is one of the two challenge messages of the adversary and $\vec{a}$ is the encoding randomness, can also be replaced with a uniformly random value, since $\vec{a}$ is uniformly random and α_i is non-zero.

This means that if all H are degenerate, i.e. the polynomials always return 0, the oracle response is useless for the adversary. Otherwise, if there is at least one non-degenerate polynomial, then we can bound the probability that the adversarially-chosen H (of degree at most 2^L) outputs 0 on independent, uniformly random input $\vec{r}$ via the Schwartz-Zippel lemma.

5 Application and evaluation

Consider a client who holds a secret vector $\vec{u} \in \mathbb{F}^n$ and wants to obtain the matrix-vector product $\vec{v} = \mathbf{A}\vec{u} \in \mathbb{F}^m$ for a large public matrix $\mathbf{A} \in \mathbb{F}^{m \times n}$. The client may lack sufficient storage to locally store this matrix $\mathbf{A}$ or the computational resources to evaluate the product efficiently. It is therefore natural to delegate the storage and computation to a (potentially untrusted) powerful server. Such a delegated verifiable computation is a core building block in many privacy-preserving protocols, including secure neural network inference [50, 63] where $\mathbf{A}$ represents a weight matrix of a linear layer, private information retrieval (PIR) [1, 46, 30] where $\mathbf{A}$ encodes the database, and oblivious ciphertext compression/decompression [41] where $\mathbf{A}$ is the public compression matrix.

For practical viability, the delegated computation should satisfy two key requirements: *privacy* and *integrity*. Homomorphic encryption provides privacy: the client encrypts $\vec{u}$ and sends $\text{Enc}(\vec{u})$ to the server, which homomorphically computes $\text{Enc}(\vec{v}) = \mathbf{A} \cdot \text{Enc}(\vec{u})$ using a PC-MV. Our goal is to additionally provide integrity, thereby transforming PC-MV into a vCOED protocol. In this section, we achieve this goal by combining our HERE instantiation for linear functions (i.e. $d = 1$), with blind proofs constructed from suitable VC protocols. We evaluate the resulting construction against vHE-based vCOED protocols and demonstrate it achieves better performance.

5.1 Our vCOED protocols for matrix-vector multiplication

In our vCOED construction with strong client privacy, we obviously evaluate the proving procedure of a VC scheme for the homomorphic operations of our encoding. Note that evaluating the linear function $f(\vec{u}) = \mathbf{A} \cdot \vec{u}$ over the encodings $\vec{c}_{u,1}, \vec{c}_{u,2} \in \mathbb{F}^n$ under $\text{sk}_{\mathcal{E},1}$ and $\text{sk}_{\mathcal{E},2}$ only amounts to evaluating f separately on the first and second component of the encoding, i.e.

$$\vec{ct}_{v,1} = f(\vec{ct}_{u,1}) = \mathbf{A} \cdot \vec{ct}_{u,1} \quad \text{and} \quad \vec{ct}_{v,2} = f(\vec{ct}_{u,2}) = \mathbf{A} \cdot \vec{ct}_{u,2}.$$

Therefore, we just need a VC scheme for the same linear function f when proving correct evaluation on the encoding level. Moreover, we only compute it twice: on $\vec{ct}_{u,1}$ and on $\vec{ct}_{u,2}$ under the corresponding key. We pick two VC schemes for

the matrix-vector product from the literature and describe them briefly *over the plaintext space*.

Randomised linear combination check. As the first VC scheme, consider the classic protocol that uses a random linear combination [11, 12, 75]. In the setup phase, the client samples a random row vector $\mathbf{r} \leftarrow \mathbb{F}^m$, computes $\mathbf{s} = \mathbf{r} \cdot \mathbf{A} \in \mathbb{F}^n$, and stores $(\mathbf{r}, \mathbf{s})$ as a verification key. In the online phase, for any input $\vec{u}$, the output $\vec{v}$ obtained via delegation is accepted if

$$\mathbf{s} \cdot \vec{u} = \mathbf{r} \cdot \vec{v}.$$

Soundness of this protocol follows directly from the Schwartz-Zippel lemma. The setup cost can be amortized over multiple queries. However, the client must compute and store a secret verification key. Combining the blind proof for this VC protocol and our linear HERE instantiation yields vCOEDMVI, a vCOED protocol for matrix-vector multiplication. A formal description of the resulting protocol is provided in Appendix C.2.

Sumcheck-based SNARK. The second VC scheme reduces proving $\vec{v} = \mathbf{A}\vec{u}$ to a sum-check instance. We represent the matrix $\mathbf{A} \in \mathbb{F}^{m \times n}$ and vectors $\vec{u}, \vec{v}$ as multilinear functions over boolean hypercubes, and let $C_{\mathbf{A}}$ denote a transparent PCS commitment of $\mathbf{A}$ to its multilinear extension published by the owner of $\mathbf{A}$. Then the function

$$f(\mathbf{x}) = \sum_{\mathbf{y} \in \{0,1\}^n} \mathbf{A}(\mathbf{x}, \mathbf{y}) \cdot u(\mathbf{y}) - v(\mathbf{x}),$$

evaluates to zero at all points in the boolean hypercube if and only if $\vec{v} = \mathbf{A}\vec{u}$. By applying the Schwartz-Zippel lemma, it suffices to check that its multilinear extension evaluates to zero at a random point τ . The resulting claim reduces to a sum-check instance over a low-degree polynomial $\mathcal{G}$, as detailed in Appendix C.1. Combining the blind proof for this VC protocol and our linear HERE instantiation yields vCOEDMVII, another vCOED protocol for matrix-vector multiplication. A formal description of the resulting protocol is provided in Appendix C.3.

Our combined vCOED protocol. The random linear combination check offers a proving procedure with no overhead since the output $\vec{v}$ is directly verified by the client. However, its setup cost is high, since the client needs to perform one matrix-vector product locally. On the other hand, the SNARK has a more costly proving phase but a low client-side setup cost.

We combine the two resulting vCOED protocols to obtain the best of both worlds. The client first uses the SNARK-based vCOED scheme to outsource the computation of $\mathbf{s} = \mathbf{r} \cdot \mathbf{A}$ in the setup phase. Then, the client keeps $(\mathbf{r}, \mathbf{s})$ as verification key and uses the vCOED scheme based on the random linear combination check for all subsequent delegations.⁸ This way, the client's work is sublinear in

⁸ The amortization here concerns only the plaintext VC verification key $(\mathbf{r}, \mathbf{s})$, which can be reused for computations involving the same matrix $\mathbf{A}$. It does not require

Table 1: Cost overview for verifiable matrix-vector product $\mathbf{A} \cdot \vec{u}$ where $\mathbf{A} \in \mathbb{F}^{m \times n}$.

Scheme	One-time setup		Online	
	Server	Client	Server	Client
vCOEDMVI	—	$\mathcal{O}(mn)$	$\mathcal{O}(mn)$	$\mathcal{O}(m+n)$
vCOEDMVII	-	-	$\mathcal{O}(mn \log(mn))$	$\mathcal{O}(m+n)^a$
combined	$\mathcal{O}(mn \log(mn))$	$\mathcal{O}(m+n)$	$\mathcal{O}(mn)$	$\mathcal{O}(m+n)$

^a The client verification is $\mathcal{O}(\log(mn))$, which is asymptotically dominated by the $\mathcal{O}(m+n)$ cost of encrypting the input vector and decrypting the output vector.

the setup phase, and the server does not generate an additional SNARK proof in the online phase. The resulting protocol is detailed in Appendix C.4. We compare the asymptotic costs of all three protocols in Table 1.

5.2 Implementation and results

Experimental setup. We provide prototype implementations⁹ in Rust using the Fheanor library [64]. All experiments are executed on an Apple Mac mini M4 Pro with 64GB RAM running macOS Sequoia 15.3.1. Our implementation is single-core, but the protocols are highly parallelizable and could benefit substantially from parallel execution. Since Fheanor is still under active development, our reported timings do not yet reflect the highest attainable performance.

Our implementation is based on the GBFV scheme [37], which supports SIMD encoding, large plaintext spaces with low noise growth. To provide sufficient IND-CPA-E security, we select a 128-bit plaintext space $\mathcal{P} = \mathbb{F}_p$ for prime $p = 248^{16} + 1$. Thus we choose $\mathbb{Z}[X]/(X^{2^N} + 1, X^{2^{(8+k)}} - 248^{2^k})$ for $k \in \{0, 1, 2\}$ as GBFV plaintext space and $\mathbb{Z}_q[X]/(X^{2^N} + 1)$ as ciphertext space, where N is the lattice dimension and q is the ciphertext modulus. Notice that while Fig. 4 assumes an HE scheme with only one slot, our implementation uses a straightforward optimization that packs 2^{8+k} encoding elements into one HE ciphertext (for $N = 2^{12}$). We target 100-bit security for all experiments.

Implementation. To compute the PC-MV, we use the Halevi-Shoup method [44] over block matrices. For vCOEDMVI, the prover computation matches exactly the PC-MV computation. In other words, the proof overhead is 2 by definition (the only overhead comes from HERE encoding). Since the homomorphic depth is low, we can choose small HE parameters. We use $N = 2^{12}$, $k = 1$ and $\log_2(q) = 110$ for a 4 MB matrix; and $N = 2^{12}$, $k = 2$ and $\log_2(q) = 135$ for

reusing the HE keys: fresh HE key pairs can be generated for each delegation task, as required by our one-time-verification security model. HE-key reuse is discussed separately in Remark 3.

⁹ <https://github.com/jannikspiessens/HERE>

matrices of 16 MB and 64 MB. Before sending the response back to the client, the server modswitches down to $\log_2(q) = 30$ to decrease the response size.

For vCOEDMVII, we implement the blind proof for the linear-time sum-check algorithm [26], applying plaintext-ciphertext multiplications and slot rotations when needed. vCOEDMVII uses the blind proof construction described in [34], i.e. we apply the BCS transformation to blind interactive oracle proofs. We set $N = 2^{13}$ and $\log_2(q) = 220$. The ciphertexts in the blind proof are modswitched down to $\log_2(q) = 30$ to decrease the response size¹⁰. We instantiate the transparent PCS using BaseFold and adopt parameters from its implementation for a similar 128-bit field setting.

Results. We report the performance of our protocols in Table 2. Our combined vCOED protocol for matrix-vector multiplication has online proof overhead (the ratio of **Compute** time in vCOED to a single PC-MV runtime) as low as $2\times$, and the response size is around 82 kB for delegating MVM for a 4 MB matrix $\mathbf{A}$. The proof overhead of vCOEDMVII is approximately $300\times$; however, this cost only occurs during the offline setup of the combined protocol and can be amortized over multiple queries. For the same matrix size of 4 MB, vCOEDMVII has a response size of approximately 679 kB, including the ciphertexts used to derive the Fiat-Shamir challenges.

5.3 Comparison

Comparison to vHE approaches. The works [75] and [73] follow the vHE approach, proving the correctness of linear operations on RLWE ciphertexts. When multiplying the matrix $\mathbf{A}$ with ciphertexts of $\vec{u} \in \mathbb{F}^n$, the vector $\vec{u}$ needs to be encoded into the HE plaintext space. Our implementation encodes $\vec{u}$ into SIMD slots with pack size of, e.g. $P = 512$, so that 512 vector elements are encoded into a single plaintext polynomial. However, such SIMD packing is not directly compatible with [75, 73], since evaluating the matrix-vector multiplication would then require automorphisms with non-linear maintenance operations which are known to be particularly expensive to prove in vHE [3]. In contrast, our HERE + blind VC approach decouples HE ciphertext evaluations from the relation being proven, so key-switching operations such as automorphisms incur no additional proof complexity.

The work [75] does not specify a concrete encoding strategy, making a direct comparison difficult in our setting, which involves multiplication over integers. The more recent work [73] instead adopts coefficient-wise encoding of $\vec{u}$ to encode PC-MV over small integers such as $\mathbb{Z}_{16}$, whereas our construction uses SIMD packing over a 128-bit integer. The encoding in [73] avoids automorphisms, but the output ciphertexts are not packed, leading to a large response

¹⁰ This includes 8 ciphertexts for the first 8 rounds and 3 ciphertexts for the last round since there we do not pack the 3 evaluations into one. From our estimation, it is possible to further ringswitch the first 8 ciphertexts down to $N = 2^{12}$, which leads to a response size of 433 kB for vCOEDMVII.

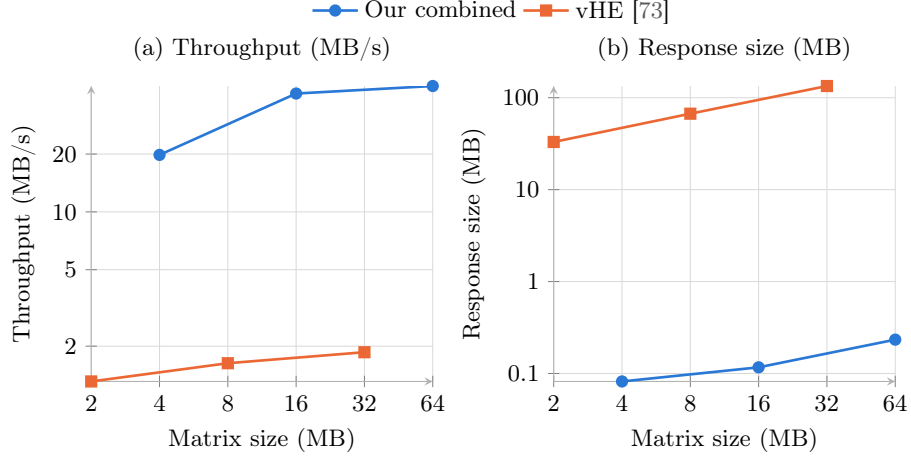


Fig. 5: An indicative comparison of our combined vCOED protocol and the vHE protocol of [73] at 100-bit security. The concrete settings and data are reported in Table 2. Both axes are log-scale.

size. Since the two systems differ in entry size, we can only perform an indicative comparison for matrices of similar sizes in Table 2. Our combined protocol gives at least $35\times$ smaller proof overhead and more than $400\times$ smaller response size. We visualize the performance comparison between our combined vCOED protocol and the vHE protocol in [73] in Fig. 5.

Comparison to verifiable PIR protocols. Verifiable PIR (vPIR) [23, 15, 68] is a specialised variant of PC-MV, where the input vector is a fresh encryption of a unit vector. This restriction allows the randomness for all input ciphertexts to be fixed in advance, which is exploited during the preprocessing phase. Consequently, vPIR protocols achieve substantially higher throughput than general-purpose PC-MV protocols while supporting efficient verification via SIS-based proofs of knowledge. In particular, the vPIR protocol in [68] achieves two to three orders of magnitude higher throughput than general-purpose PC-MV.

In contrast, our vCOED protocol has $2\times$ lower throughput than a general-purpose PC-MV, but provides integrity for arbitrary query vectors and does not require part of the ciphertexts to be fixed in advance. Moreover, general-purpose PC-MV can serve as a building block for broader applications. While this section only focuses on vCOED protocols for MVM, our approach can be extended to support pre-processing around MVM, such as deriving the predicate input vector [10], and post-processing, such as computing statistics over the output vector [31]. Such an extension is possible with our HERE instantiation for $d > 1$ together with suitable blind-proof protocols.

Table 2: Comparison of concrete performance for the online computation in verifiable matrix-vector product $\mathbf{A} \cdot \vec{u}$ where $\mathbf{A} \in \mathbb{F}^{m \times m}$. All of our parameter sets provide at least 100-bit security. Columns: “Proof overhead” is the ratio between server runtime with and without integrity, i.e. the run time in vCOED (PC-MV+proof) divided by the runtime of a single PC-MV under the same HE parameters; “Throughput” is the ratio of the matrix $\mathbf{A}$ size to the server runtime; “Response size” is the total size of communication from server back to client including proof.

Scheme	Matrix size	m	Proof overhead	Throughput (MB/s)	Response size
vCOEDMVI	4 MB	2^9	2	19.8 (1 thread)	81.9 kB
vCOEDMVII	4 MB	2^9	300	0.023 (1 thread)	679 kB
combined	4 MB	2^9	2	19.8 (1 thread)	81.9 kB
combined	16 MB	2^{10}	2	41.3 (1 thread)	116.7 kB
combined	64 MB	2^{11}	2	45.2 (1 thread)	233.4 kB
vHE in [73]	2 MB	2^{11}	$\frac{1.522s}{0.017s} \approx 90$	1.31 (14 threads)	33 MB
vHE in [73]	8 MB	2^{12}	$\frac{4.914s}{0.066s} \approx 74$	1.63 (14 threads)	67 MB
vHE in [73]	32 MB	2^{13}	$\frac{17.23s}{0.242s} \approx 71$	1.86 (14 threads)	134 MB

6 Conclusion and future direction

We introduce HERE and its associated IND-CPA-E security notion. We instantiate HERE for bounded-degree functions using Shamir secret sharing, and prove its IND-CPA-E security under the multi-key, bounded-depth circuit-awareness assumption. Combining HERE with blind proofs yields vCOED schemes that are correct, sound, and secure in the sense of strong client privacy. As an application, we build vCOED protocols for matrix-vector multiplication that follow this paradigm and achieve only $2\times$ proof overhead.

Future directions. Possible theoretical future directions include the following. First, our HERE encoding has $(d + 1)$ components to support evaluation of a degree- d function. One direction for future work is to construct HERE encodings whose size grows sublinearly with the polynomial degree. Second, our construction requires a large encoding space of $|\mathbb{F}| \geq 2^\lambda$ for λ -bit security. Therefore, it would be interesting to construct HERE that supports smaller fields.

On the application side, applying our HERE instantiation to build vCOED for non-linear functions is a promising direction. Moreover, while our vCOED for MVM can naively be extended to matrix-matrix multiplication, the resulting protocol does not use packing methods in [49, 4, 65, 40] that facilitate homomorphic evaluation. We leave it as future work to exploit such packing strategies and build efficient vCOED protocols for matrix-matrix multiplication.

Acknowledgments. This work is partly supported by Australian Research Council Discovery Grant DP250100229 and CyberSecurity Research Flanders with reference number VOEWICS02. Jannik Spiessens is funded by an FWO fellowship with reference number 1139125N. The authors thank Frederik Vercauteren, Robin Geelen, and the anonymous reviewers for their helpful feedback on this paper. The authors would also like to thank Simon Pohmann for brainstorming during the initial stage of this project, and Antonio Guimarães, Muhammed Esgin, Wangfan Gu, Amin Sakzad, Ron Steinfeld for useful discussions on key-recovery attacks and for bringing several related works to our attention.

References

1. Aguilar-Melchor, C., Barrier, J., Fousse, L., Killijian, M.O.: XPIR: Private information retrieval for everyone. *PoPETS* **2016**(2), 155–174 (Apr 2016). <https://doi.org/10.1515/popets-2016-0010>
2. Aranha, D.F., Costache, A., Guimarães, A., Soria-Vazquez, E.: HELIOPOLIS: Verifiable computation over homomorphically encrypted data from interactive oracle proofs is practical. In: Chung, K.M., Sasaki, Y. (eds.) *ASIACRYPT 2024*, Part V. LNCS, vol. 15488, pp. 302–334. Springer, Singapore (Dec 2024). https://doi.org/10.1007/978-981-96-0935-2_10
3. Atapoor, S., Baghery, K., Pereira, H.V.L., Spiessens, J.: Verifiable FHE via lattice-based SNARKs. *CiC* **1**(1), 24 (2024). <https://doi.org/10.62056/a6ksdkp10>
4. Bae, Y., Cheon, J.H., Hanrot, G., Park, J.H., Stehlé, D.: Plaintext-ciphertext matrix multiplication and FHE bootstrapping: Fast and fused. In: Reyzin, L., Stebila, D. (eds.) *CRYPTO 2024*, Part III. LNCS, vol. 14922, pp. 387–421. Springer, Cham (Aug 2024). https://doi.org/10.1007/978-3-031-68382-4_12
5. Bellare, M., Desai, A., Pointcheval, D., Rogaway, P.: Relations among notions of security for public-key encryption schemes. In: Krawczyk, H. (ed.) *CRYPTO’98*. LNCS, vol. 1462, pp. 26–45. Springer, Berlin, Heidelberg (Aug 1998). <https://doi.org/10.1007/BFb0055718>
6. Bellare, M., Palacio, A.: Towards plaintext-aware public-key encryption without random oracles. In: Lee, P.J. (ed.) *ASIACRYPT 2004*. LNCS, vol. 3329, pp. 48–62. Springer, Berlin, Heidelberg (Dec 2004). https://doi.org/10.1007/978-3-540-30539-2_4
7. Bellare, M., Rogaway, P.: Optimal asymmetric encryption. In: De Santis, A. (ed.) *EUROCRYPT’94*. LNCS, vol. 950, pp. 92–111. Springer, Berlin, Heidelberg (May 1995). <https://doi.org/10.1007/BFb0053428>
8. Ben-Sasson, E., Chiesa, A., Spooner, N.: Interactive oracle proofs. In: Hirt, M., Smith, A.D. (eds.) *TCC 2016-B*, Part II. LNCS, vol. 9986, pp. 31–60. Springer, Berlin, Heidelberg (Oct / Nov 2016). https://doi.org/10.1007/978-3-662-53644-5_2
9. Bitansky, N., Chiesa, A., Ishai, Y., Ostrovsky, R., Paneth, O.: Succinct non-interactive arguments via linear interactive proofs. In: Sahai, A. (ed.) *TCC 2013*. LNCS, vol. 7785, pp. 315–333. Springer, Berlin, Heidelberg (Mar 2013). https://doi.org/10.1007/978-3-642-36594-2_18
10. Blindenbach, J., Kang, J., Hong, S., Karam, C., Lehner, T., Gürsoy, G.: SQuID: ultra-secure storage and analysis of genetic data for the advancement of precision medicine. *Genome Biology* **25**(314) (2024). <https://doi.org/10.1186/s13059-024-03447-9>

11. Blum, M., Luby, M., Rubinfeld, R.: Self-testing/correcting with applications to numerical problems. In: 22nd ACM STOC. pp. 73–83. ACM Press (May 1990). <https://doi.org/10.1145/100216.100225>
12. Blum, M., Wasserman, H.: Program result-checking: A theory of testing meets a test of theory. In: 35th FOCS. pp. 382–392. IEEE Computer Society Press (Nov 1994). <https://doi.org/10.1109/SFCS.1994.365678>
13. Brakerski, Z., Gentry, C., Vaikuntanathan, V.: (Leveled) fully homomorphic encryption without bootstrapping. In: Goldwasser, S. (ed.) ITCS 2012. pp. 309–325. ACM (Jan 2012). <https://doi.org/10.1145/2090236.2090262>
14. Bünz, B., Fisch, B., Szepieniec, A.: Transparent SNARKs from DARK compilers. In: Canteaut, A., Ishai, Y. (eds.) EUROCRYPT 2020, Part I. LNCS, vol. 12105, pp. 677–706. Springer, Cham (May 2020). https://doi.org/10.1007/978-3-030-45721-1_24
15. de Castro, L., Lee, K.: VeriSimplePIR: Verifiability in SimplePIR at no on-line cost for honest servers. In: Balzarotti, D., Xu, W. (eds.) USENIX Security 2024. USENIX Association (Aug 2024), <https://www.usenix.org/conference/usenixsecurity24/presentation/de-castro>
16. Catalano, D., Fiore, D.: Practical homomorphic MACs for arithmetic circuits. In: Johansson, T., Nguyen, P.Q. (eds.) EUROCRYPT 2013. LNCS, vol. 7881, pp. 336–352. Springer, Berlin, Heidelberg (May 2013). https://doi.org/10.1007/978-3-642-38348-9_21
17. Chatel, S., Knabenhans, C., Pyrgelis, A., Troncoso, C., Hubaux, J.P.: VERITAS: Plaintext encoders for practical verifiable homomorphic encryption. In: Luo, B., Liao, X., Xu, J., Kirda, E., Lie, D. (eds.) ACM CCS 2024. pp. 2520–2534. ACM Press (Oct 2024). <https://doi.org/10.1145/3658644.3670282>
18. Checri, M., Sirdey, R., Boudguiga, A., Bultel, J.P.: On the practical CPA^D security of “exact” and threshold FHE schemes and libraries. In: Reyzin, L., Stebila, D. (eds.) CRYPTO 2024, Part III. LNCS, vol. 14922, pp. 3–33. Springer, Cham (Aug 2024). https://doi.org/10.1007/978-3-031-68382-4_1
19. Chenal, M., Tang, Q.: On key recovery attacks against existing somewhat homomorphic encryption schemes. In: Aranha, D.F., Menezes, A. (eds.) LATIN-CRYPT 2014. LNCS, vol. 8895, pp. 239–258. Springer, Cham (Sep 2015). https://doi.org/10.1007/978-3-319-16295-9_13
20. Cheon, J.H., Choe, H., Passelègue, A., Stehlé, D., Suvanto, E.: Attacks against the IND-CPA^D security of exact FHE schemes. In: Luo, B., Liao, X., Xu, J., Kirda, E., Lie, D. (eds.) ACM CCS 2024. pp. 2505–2519. ACM Press (Oct 2024). <https://doi.org/10.1145/3658644.3690341>
21. Cheon, J.H., Jang, D.: Cryptanalysis on lightweight verifiable homomorphic encryption. In: Hanaoka, G., Yang, B.Y. (eds.) ASIACRYPT 2025, Part VII. LNCS, vol. 16251, pp. 366–397. Springer, Singapore (Dec 2025). https://doi.org/10.1007/978-981-95-5122-4_12
22. Chillotti, I., Gama, N., Goubin, L.: Attacking FHE-based applications by software fault injections. Cryptology ePrint Archive, Report 2016/1164 (2016), <https://eprint.iacr.org/2016/1164>
23. Colombo, S., Nikitin, K., Corrigan-Gibbs, H., Wu, D.J., Ford, B.: Authenticated private information retrieval. In: Calandrino, J.A., Troncoso, C. (eds.) USENIX Security 2023. pp. 3835–3851. USENIX Association (Aug 2023), <https://www.usenix.org/conference/usenixsecurity23/presentation/colombo>
24. Cong, K., Das, D., Park, J., Pereira, H.V.L.: SortingHat: Efficient private decision tree evaluation via homomorphic encryption and transciphering. In: Yin, H.,

- Stavrou, A., Cremers, C., Shi, E. (eds.) ACM CCS 2022. pp. 563–577. ACM Press (Nov 2022). <https://doi.org/10.1145/3548606.3560702>
25. Cong, K., Kang, J., Nicolas, G., Park, J.: Faster private decision tree evaluation for batched input from homomorphic encryption. In: Galdi, C., Phan, D.H. (eds.) SCN 24, Part II. LNCS, vol. 14974, pp. 3–23. Springer, Cham (Sep 2024). https://doi.org/10.1007/978-3-031-71073-5_1
 26. Cormode, G., Mitzenmacher, M., Thaler, J.: Practical verified computation with streaming interactive proofs. In: Goldwasser, S. (ed.) ITCS 2012. pp. 90–112. ACM (Jan 2012). <https://doi.org/10.1145/2090236.2090245>
 27. Dachman-Soled, D., Ducas, L., Gong, H., Rossi, M.: LWE with side information: Attacks and concrete security estimation. In: Micciancio, D., Ristenpart, T. (eds.) CRYPTO 2020, Part II. LNCS, vol. 12171, pp. 329–358. Springer, Cham (Aug 2020). https://doi.org/10.1007/978-3-030-56880-1_12
 28. Dachman-Soled, D., Gong, H., Hanson, T., Kippen, H.: Revisiting security estimation for LWE with hints from a geometric perspective. In: Handschuh, H., Lysyanskaya, A. (eds.) CRYPTO 2023, Part V. LNCS, vol. 14085, pp. 748–781. Springer, Cham (Aug 2023). https://doi.org/10.1007/978-3-031-38554-4_24
 29. Dao, Q., Grubbs, P.: Spartan and bulletproofs are simulation-extractable (for free!). In: Hazay, C., Stam, M. (eds.) EUROCRYPT 2023, Part II. LNCS, vol. 14005, pp. 531–562. Springer, Cham (Apr 2023). https://doi.org/10.1007/978-3-031-30617-4_18
 30. Davidson, A., Pestana, G., Celi, S.: FrodoPIR: Simple, scalable, single-server private information retrieval. PoPETs **2023**(1), 365–383 (Jan 2023). <https://doi.org/10.56553/popets-2023-0022>
 31. Fiore, D., Gennaro, R., Pastro, V.: Efficiently verifiable computation on encrypted data. In: Ahn, G.J., Yung, M., Li, N. (eds.) ACM CCS 2014. pp. 844–855. ACM Press (Nov 2014). <https://doi.org/10.1145/2660267.2660366>
 32. Fiore, D., Nitulescu, A., Pointcheval, D.: Boosting verifiable computation on encrypted data. In: Kiayias, A., Kohlweiss, M., Wallden, P., Zikas, V. (eds.) PKC 2020, Part II. LNCS, vol. 12111, pp. 124–154. Springer, Cham (May 2020). https://doi.org/10.1007/978-3-030-45388-6_5
 33. Fisch, B., Liu, Z., Tromer, E., Wang, Y.: ÜnifOMR: Oblivious message retrieval with near-optimal concrete efficiency. Cryptology ePrint Archive, Paper 2026/910 (2026), <https://eprint.iacr.org/2026/910>
 34. Gama, M., Beni, E.H., Kang, J., Spiessens, J., Vercauteren, F.: Blind zkSNARKs: for private proof delegation and verifiable computation over encrypted data. CiC **2**(3), 10 (2025). <https://doi.org/10.62056/akgyl8n4e>
 35. Ganesh, C., Orlandi, C., Pancholi, M., Takahashi, A., Tschudi, D.: Fiat–shamir bulletproofs are non-malleable (in the random oracle model). Journal of Cryptology **38**(1), 11 (2024). <https://doi.org/10.1007/s00145-024-09525-2>
 36. Garg, S., Goel, A., Wang, M.: How to prove statements obliviously? In: Reyzin, L., Stebila, D. (eds.) CRYPTO 2024, Part X. LNCS, vol. 14929, pp. 449–487. Springer, Cham (Aug 2024). https://doi.org/10.1007/978-3-031-68403-6_14
 37. Geelen, R., Vercauteren, F.: Fully homomorphic encryption for cyclotomic prime moduli. In: Fehr, S., Fouque, P.A. (eds.) EUROCRYPT 2025, Part III. LNCS, vol. 15603, pp. 366–397. Springer, Cham (May 2025). https://doi.org/10.1007/978-3-031-91131-6_13
 38. Gennaro, R., Gentry, C., Parno, B.: Non-interactive verifiable computing: Outsourcing computation to untrusted workers. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 465–482. Springer, Berlin, Heidelberg (Aug 2010). https://doi.org/10.1007/978-3-642-14623-7_25

39. Gennaro, R., Wichs, D.: Fully homomorphic message authenticators. In: Sako, K., Sarkar, P. (eds.) ASIACRYPT 2013, Part II. LNCS, vol. 8270, pp. 301–320. Springer, Berlin, Heidelberg (Dec 2013). https://doi.org/10.1007/978-3-642-42045-0_16
40. Gentry, C., Lee, Y.: Fully homomorphic encryption for matrix arithmetic. In: Heninger, N., Rosulek, M. (eds.) Advances in Cryptology - CRYPTO 2026 - 46th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 17–20, 2026, Proceedings, Part II. Lecture Notes in Computer Science, vol. 16801, pp. 363–395. Springer (2026), https://doi.org/10.1007/978-3-032-35374-0_12
41. Giorgi, P., Grenet, B., Simkin, M.: Oblivious ciphertext compression via linear codes. In: Daemen, J., Thomé, E. (eds.) Advances in Cryptology – EUROCRYPT 2026. pp. 366–391. Springer Nature Switzerland, Cham (2026)
42. Goldreich, O.: On doubly-efficient interactive proof systems. *Found. Trends Theor. Comput. Sci.* **13**(3), 158–246 (Apr 2018). <https://doi.org/10.1561/04000000084>
43. Golovnev, A., Lee, J., Setty, S.T.V., Thaler, J., Wahby, R.S.: Brakedown: Linear-time and field-agnostic SNARKs for R1CS. In: Handschuh, H., Lysyanskaya, A. (eds.) CRYPTO 2023, Part II. LNCS, vol. 14082, pp. 193–226. Springer, Cham (Aug 2023). https://doi.org/10.1007/978-3-031-38545-2_7
44. Halevi, S., Shoup, V.: Faster homomorphic linear transformations in HELib. In: Shacham, H., Boldyreva, A. (eds.) CRYPTO 2018, Part I. LNCS, vol. 10991, pp. 93–120. Springer, Cham (Aug 2018). https://doi.org/10.1007/978-3-319-96884-1_4
45. Hall, C., Goldberg, I., Schneier, B.: Reaction attacks against several public-key cryptosystems. In: Varadharajan, V., Mu, Y. (eds.) ICICS 99. LNCS, vol. 1726, pp. 2–12. Springer, Berlin, Heidelberg (Nov 1999). https://doi.org/10.1007/978-3-540-47942-0_2
46. Henzinger, A., Hong, M.M., Corrigan-Gibbs, H., Meiklejohn, S., Vaikuntanathan, V.: One server for the price of two: Simple and fast single-server private information retrieval. In: Calandrino, J.A., Troncoso, C. (eds.) USENIX Security 2023. pp. 3889–3905. USENIX Association (Aug 2023), <https://www.usenix.org/conference/usenixsecurity23/presentation/henzinger>
47. Hu, Z., Sun, F., Jiang, J.: Ciphertext verification security of symmetric encryption schemes. *Science in China Series F: Information Sciences* **52**(9), 1617–1631 (2009)
48. Ishai, Y., Su, H., Wu, D.J.: Shorter and faster post-quantum designated-verifier zkSNARKs from lattices. In: Vigna, G., Shi, E. (eds.) ACM CCS 2021. pp. 212–234. ACM Press (Nov 2021). <https://doi.org/10.1145/3460120.3484572>
49. Jiang, X., Kim, M., Lauter, K.E., Song, Y.: Secure outsourced matrix computation and application to neural networks. In: Lie, D., Mannan, M., Backes, M., Wang, X. (eds.) ACM CCS 2018. pp. 1209–1222. ACM Press (Oct 2018). <https://doi.org/10.1145/3243734.3243837>
50. Juvekar, C., Vaikuntanathan, V., Chandrakasan, A.: GAZELLE: A low latency framework for secure neural network inference. In: Enck, W., Felt, A.P. (eds.) USENIX Security 2018. pp. 1651–1669. USENIX Association (Aug 2018), <https://www.usenix.org/conference/usenixsecurity18/presentation/juvekar>
51. Kang, J.: Optimisations and Applications of Fully Homomorphic Encryption. Phd thesis, KU Leuven (November 2025), <https://www.esat.kuleuven.be/cosic/education/finished-phd-master-theses/>
52. Knabenhans, C., Viand, A., Merino-Gallardo, A., Hithnawi, A.: vfhe: Verifiable fully homomorphic encryption. In: Bergamaschi, F., Costache, A., Rohloff, K. (eds.) Proceedings of the 12th Workshop on Encrypted Computing & Applied

- Homomorphic Cryptography, Salt Lake City, UT, USA, October 14-18, 2024. pp. 11–22. ACM (2024). <https://doi.org/10.1145/3689945.3694806>
53. Kohlweiss, M., Pancholi, M., Takahashi, A.: How to compile polynomial IOP into simulation-extractable SNARKs: A modular approach. In: Rothblum, G.N., Wee, H. (eds.) TCC 2023, Part III. LNCS, vol. 14371, pp. 486–512. Springer, Cham (Nov / Dec 2023). https://doi.org/10.1007/978-3-031-48621-0_17
 54. Lai, J., Deng, R.H., Pang, H., Weng, J.: Verifiable computation on outsourced encrypted data. In: Kutyłowski, M., Vaidya, J. (eds.) ESORICS 2014, Part I. LNCS, vol. 8712, pp. 273–291. Springer, Cham (Sep 2014). https://doi.org/10.1007/978-3-319-11203-9_16
 55. Li, B., Micciancio, D.: On the security of homomorphic encryption on approximate numbers. In: Canteaut, A., Standaert, F.X. (eds.) EUROCRYPT 2021, Part I. LNCS, vol. 12696, pp. 648–677. Springer, Cham (Oct 2021). https://doi.org/10.1007/978-3-030-77870-5_23
 56. Li, S., Wang, X., Zhang, R.: Privacy-preserving homomorphic macs with efficient verification. In: Jin, H., Wang, Q., Zhang, L. (eds.) Web Services - ICWS 2018 - 25th International Conference, Held as Part of the Services Conference Federation, SCF 2018, Seattle, WA, USA, June 25-30, 2018, Proceedings. pp. 100–115. Lecture Notes in Computer Science, Springer (2018). https://doi.org/10.1007/978-3-319-94289-6_7
 57. Liu, F., Liang, H., Xie, X., Yu, Y., Zheng, W., Hu, Y.: HasteBoots: Proving TFHE programmable bootstrapping in seconds. In: 35th USENIX Security Symposium (USENIX Security 26). pp. 41–60. USENIX Association, Baltimore, MD (Aug 2026), <https://www.usenix.org/conference/usenixsecurity26/presentation/liu-fengrun>
 58. Liu, Z., Tromer, E.: Oblivious message retrieval. In: Dodis, Y., Shrimpton, T. (eds.) CRYPTO 2022, Part I. LNCS, vol. 13507, pp. 753–783. Springer, Cham (Aug 2022). https://doi.org/10.1007/978-3-031-15802-5_26
 59. Liu, Z., Tromer, E., Wang, Y.: PerfOMR: Oblivious message retrieval with reduced communication and computation. In: Balzarotti, D., Xu, W. (eds.) USENIX Security 2024. USENIX Association (Aug 2024), <https://www.usenix.org/conference/usenixsecurity24/presentation/liu-zeyu>
 60. Loftus, J., May, A., Smart, N.P., Vercauteren, F.: On CCA-secure somewhat homomorphic encryption. In: Miri, A., Vaudenay, S. (eds.) SAC 2011. LNCS, vol. 7118, pp. 55–72. Springer, Berlin, Heidelberg (Aug 2012). https://doi.org/10.1007/978-3-642-28496-0_4
 61. Lund, C., Fortnow, L., Karloff, H.J., Nisan, N.: Algebraic methods for interactive proof systems. In: 31st FOCS. pp. 2–10. IEEE Computer Society Press (Oct 1990). <https://doi.org/10.1109/FSCS.1990.89518>
 62. Mahdavi, R.A., Ni, H., Linkov, D., Kerschbaum, F.: Level up: Private non-interactive decision tree evaluation using levelled homomorphic encryption. In: Meng, W., Jensen, C.D., Cremers, C., Kirda, E. (eds.) ACM CCS 2023. pp. 2945–2958. ACM Press (Nov 2023). <https://doi.org/10.1145/3576915.3623095>
 63. Mann, Z.Á., Weinert, C., Chabal, D., Bos, J.W.: Towards practical secure neural network inference: The journey so far and the road ahead. ACM Comput. Surv. **56**(5), 117:1–117:37 (2024). <https://doi.org/10.1145/3628446>
 64. Okada, H., Player, R., Pohmann, S.: Fheanor: a new, modular FHE library for designing and optimising schemes. Cryptology ePrint Archive, Report 2025/864 (2025), <https://eprint.iacr.org/2025/864>

65. Park, J.H.: Ciphertext-ciphertext matrix multiplication: Fast for large matrices. In: Fehr, S., Fouque, P.A. (eds.) EUROCRYPT 2025, Part VIII. LNCS, vol. 15608, pp. 153–180. Springer, Cham (May 2025). https://doi.org/10.1007/978-3-031-91101-9_6
66. Parno, B., Howell, J., Gentry, C., Raykova, M.: Pinocchio: Nearly practical verifiable computation. In: 2013 IEEE Symposium on Security and Privacy. pp. 238–252. IEEE Computer Society Press (May 2013). <https://doi.org/10.1109/SP.2013.47>
67. Peshawaria, K., Liu, Z., Fisch, B., Tromer, E.: Laminar: Succinct SIMD-friendly verifiable FHE. Cryptology ePrint Archive, Report 2025/2285 (2025), <https://eprint.iacr.org/2025/2285>
68. Rathee, M., Lee, K., Popa, R.A.: Verifiable PIR with small client storage. In: 2026 IEEE Symposium on Security and Privacy. pp. 1423–1442. IEEE Computer Society Press, San Francisco, CA, USA (May 18–20, 2026). <https://doi.org/10.1109/SP63933.2026.00024>
69. Setty, S.: Spartan: Efficient and general-purpose zkSNARKs without trusted setup. In: Micciancio, D., Ristenpart, T. (eds.) CRYPTO 2020, Part III. LNCS, vol. 12172, pp. 704–737. Springer, Cham (Aug 2020). https://doi.org/10.1007/978-3-030-56877-1_25
70. Shamir, A.: How to share a secret. Communications of the Association for Computing Machinery **22**(11), 612–613 (Nov 1979). <https://doi.org/10.1145/359168.359176>
71. Smart, N.P., Vercauteren, F.: Fully homomorphic SIMD operations. DCC **71**(1), 57–81 (2014). <https://doi.org/10.1007/s10623-012-9720-4>
72. Thibault, L.T., Walter, M.: Towards verifiable FHE in practice: Proving correct execution of TFHE’s bootstrapping using plonky2. In: Huang, C.Y., Chen, J.C., Shieh, S.P., Lie, D., Cortier, V. (eds.) ACM CCS 2025. pp. 1113–1126. ACM Press (Oct 2025). <https://doi.org/10.1145/3719027.3765097>
73. Thibault, L.T., Walter, M., Zhang, J.: Practical SNARKs for matrix multiplications over encrypted data. Cryptology ePrint Archive, Paper 2026/027 (2026), <https://eprint.iacr.org/2026/027>
74. Zeilberger, H., Chen, B., Fisch, B.: BaseFold: Efficient field-agnostic polynomial commitment schemes from foldable codes. In: Reyzin, L., Stebila, D. (eds.) CRYPTO 2024, Part X. LNCS, vol. 14929, pp. 138–169. Springer, Cham (Aug 2024). https://doi.org/10.1007/978-3-031-68403-6_5
75. Zhang, L.F., Safavi-Naini, R.: Protecting data privacy in publicly verifiable delegation of matrix and polynomial functions. DCC **88**(4), 677–709 (2020). <https://doi.org/10.1007/s10623-019-00704-y>
76. Zhang, X., Wang, R., Liu, Z., Xiang, B., Deng, Y., Fisch, B., Lu, X.: Phalanx: An FHE-friendly SNARK for verifiable computation on encrypted data. In: Huang, C.Y., Chen, J.C., Shieh, S.P., Lie, D., Cortier, V. (eds.) ACM CCS 2025. pp. 4035–4048. ACM Press (Oct 2025). <https://doi.org/10.1145/3719027.3765226>
77. Zhang, X., Wang, R., Niu, Q., Liu, P., Lu, X., Zhao, L., Hou, R., Deng, Y.: Lasagne: Practical verifiable computation over encrypted data. Cryptology ePrint Archive, Paper 2026/857 (2026), <https://eprint.iacr.org/2026/857>
78. Zhang, Z., Plantard, T., Susilo, W.: Reaction attack on outsourced computing with fully homomorphic encryption schemes. In: Kim, H. (ed.) ICISC 11. LNCS, vol. 7259, pp. 419–436. Springer, Berlin, Heidelberg (Nov / Dec 2012). https://doi.org/10.1007/978-3-642-31912-9_28

A Definition of Circuit Extractor (Multi-Key)

Definition 13 (Circuit Extractor (multi-key)). Let $\mathcal{E}$ be a homomorphic encryption scheme, $n \in \mathbb{N}$ and $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, $\mathcal{A}^*$ and $\mathcal{D}$ be PPT algorithms. We say $\mathcal{A}$ is multi-key, L -bounded circuit-aware, if there exists a circuit extractor $\mathcal{A}^*$, s.t. for any distinguisher $\mathcal{D}$, and any random coins $r, r^* \leftarrow_{\$} \{0, 1\}^*$, the difference of probabilities $\Pr[\text{real}]$ and $\Pr[\text{extracted}]$ defined as

$$\Pr[\text{real}] = \Pr \left[\mathcal{D}(\mathcal{A}_2(\text{st}, x)) = 1 \mid \begin{array}{l} \text{sk}_i, \text{pk}_i \leftarrow \mathcal{E}.\text{KeyGen}(1^\lambda), \text{ for } i \in [n] \\ \text{ct}[x], \text{st} \leftarrow \mathcal{A}_1^{\text{O}}(1^\lambda, \{\text{pk}_i\}_{i=1}^n; r) \\ x \leftarrow \mathcal{E}.\text{Dec}_{\text{sk}_1}(\text{ct}[x]) \end{array} \right],$$

and

$$\Pr[\text{extracted}] = \Pr \left[\mathcal{D}(\mathcal{A}_2(\text{st}, x)) = 1 \mid \begin{array}{l} \text{sk}_i, \text{pk}_i \leftarrow \mathcal{E}.\text{KeyGen}(1^\lambda), \text{ for } i \in [n] \\ \text{ct}[x], \text{st} \leftarrow \mathcal{A}_1^{\text{O}}(1^\lambda, \{\text{pk}_i\}_{i=1}^n; r) \\ F \leftarrow \mathcal{A}^*(1^\lambda, \{\text{pk}_i\}_{i=1}^n, r; r^*) \\ x_j^{\text{O}} \leftarrow \mathcal{E}.\text{Dec}_{\text{sk}_1}(\text{ct}[x_j^{\text{O}}]) \text{ for } j \in [q] \\ x \leftarrow F(x_1^{\text{O}}, \dots, x_q^{\text{O}}) \end{array} \right],$$

is negligible, i.e., $\text{Adv}_{\mathcal{D}}^{\text{mk-ext}}(\lambda) = |\Pr[\text{real}] - \Pr[\text{extracted}]| \leq \text{negl}(\lambda)$, and F is a circuit of depth at most L . Above, $\text{O}(x)$ denotes an arbitrary oracle. If the j -th oracle call outputs a ciphertext to $\mathcal{A}_1$, we denote this ciphertext as $\text{ct}[x_j^{\text{O}}]$. Note that $\text{ct}[x_j^{\text{O}}]$ can be encrypted under any sk_i .

B Deferred Proofs

B.1 Proof of Theorem 3

Proof. We transform the IND-CPA-E game with adversary $\mathcal{A}$ against our HERE scheme in Fig. 4 in multiple hybrid games, defined as follows.

- Hybrid G_0 : We modify $G_{\mathcal{A}}^{\text{ind-e}}$. The call to EnTEnc is now encapsulated in the left-or-right oracle O_{LR} :

$G_0(\lambda)$	$\text{O}_{\text{LR}}(\vec{x}_0, \vec{x}_1)$
1 : $b \leftarrow_{\$} \{0, 1\}$	this oracle runs at most once
2 : $(\text{sk}, \text{pk}) \leftarrow \text{KeyGen}(1^\lambda, \mathcal{F})$	and has access to $\vec{a}$
3 : $\vec{a} \leftarrow_{\$} \mathbb{F}_p^\ell$	1 : for $i \in [d+1]$ do
4 : $(\vec{x}_0, \vec{x}_1, \text{st}_{\mathcal{A}}) \leftarrow \mathcal{A}_1^{\text{OEn}}(1^\lambda, \text{pk})$	2 : $\vec{c}_i \leftarrow \vec{x}_b + \alpha_i \cdot \vec{a}$
5 : $\vec{\text{ct}}_1^*, \dots, \vec{\text{ct}}_{d+1}^* \leftarrow \text{O}_{\text{LR}}(\vec{x}_0, \vec{x}_1)$	3 : $\vec{\text{ct}}_i^* \leftarrow \mathcal{E}.\text{Enc}_{\text{sk}_{\mathcal{E}, i}}(\vec{c}_i)$
6 : $b' \leftarrow \mathcal{A}_2^{\text{OEn}, \text{OEqual}}(1^\lambda, \text{st}_{\mathcal{A}}, \vec{\text{ct}}_1^*, \dots, \vec{\text{ct}}_{d+1}^*)$	4 : return $\vec{\text{ct}}_1^*, \dots, \vec{\text{ct}}_{d+1}^*$
7 : return 1 iff. $b = b'$	

These changes are only syntactic and do not change the advantage of $(\mathcal{A}_1, \mathcal{A}_2)$. Further, we replace the encode-then-encrypt oracle $\mathcal{O}_{\text{En}}$ with an HE encryption oracle. Note that this does not change the advantage of $(\mathcal{A}_1, \mathcal{A}_2)$: Since our encoding does not have an encoding secret key sk_1 , the adversary can sample the random polynomial itself and create the encoding. It then uses the HE encryption oracle to encrypt the encoding with the respective HE secret key. We will keep $\mathcal{O}_{\text{En}}$ in the following but it now denotes the HE encryption oracle.

- Hybrid G_1 : We modify G_0 . The first part of the adversary $\mathcal{A}_1$ now receives access to $\mathcal{O}_{\text{LR}}$ directly, and instead outputs the query CT to the equality oracle and $\text{st}_{\mathcal{A}}$, where

$$\text{CT} = \left(\text{ct}_{u,1}^{(j)}, \text{ct}_{v,1}^{(j)}, \dots, \text{ct}_{u,d+1}^{(j)}, \text{ct}_{v,d+1}^{(j)} \right)_{j=1}^q.$$

We also run $\delta \leftarrow \mathcal{O}_{\text{Equal}}(\text{CT})$ as the challenger and input δ to $\mathcal{A}_2$. G_1 is now defined as:

$$\begin{array}{l} \hline G_1(\lambda) \\ \hline 1 : \quad b \leftarrow_{\$} \{0, 1\} \\ 2 : \quad (\text{sk}, \text{pk}) \leftarrow \text{KeyGen}(1^\lambda, \mathcal{F}) \\ 3 : \quad \vec{a} \leftarrow_{\$} \mathbb{F}_p^\ell \\ 4 : \quad (\text{CT}, \text{st}_{\mathcal{A}}) \leftarrow \mathcal{A}_1^{\mathcal{O}_{\text{En}}, \mathcal{O}_{\text{LR}}}(1^\lambda, \text{pk}) \\ 5 : \quad \delta \leftarrow \mathcal{O}_{\text{Equal}}(\text{CT}) \\ 6 : \quad b' \leftarrow \mathcal{A}_2^{\mathcal{O}_{\text{En}}}(1^\lambda, \text{st}_{\mathcal{A}}, \delta) \\ 7 : \quad \textbf{return } 1 \text{ iff. } b = b' \end{array}$$

This change reorganizes the (arbitrary) partition of the algorithm $\mathcal{A}$ and is only syntactic.

For the next hybrid, we will use the circuit extractor from Definition 13, which we reproduce in adapted form, filling in all oracles that the hybrid games use. Note that in Definition 14 below, we give the extractor $\mathcal{A}^*$ an additional input ‘label’ that describes which of the ciphertexts in CT should be extracted.

Definition 14 (Circuit Extractor (multi-key), adapted). *Let $\mathcal{E}$ be a homomorphic encryption scheme, $n \in \mathbb{N}$ and $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, $\mathcal{A}^*$ and $\mathcal{D}$ be PPT algorithms. We say $\mathcal{A}$ is multi-key, L -bounded circuit-aware, if there exists a circuit extractor $\mathcal{A}^*$, s.t. for any distinguisher $\mathcal{D}$, and any random coins $r, r^* \leftarrow_{\$} \{0, 1\}^*$, the difference of probabilities $\Pr[\text{real}]$ and $\Pr[\text{extracted}]$ defined as*

$$\Pr[\text{real}] = \Pr \left[\mathcal{D}(\mathcal{A}_2(\text{st}, x)) = 1 \mid \begin{array}{l} \text{sk}_i, \text{pk}_i \leftarrow \mathcal{E}.\text{KeyGen}(1^\lambda), \text{ for } i \in [n] \\ \text{ct}[x], \text{st} \leftarrow \mathcal{A}_1^{\mathcal{O}_{\text{En}}, \mathcal{O}_{\text{LR}}}(1^\lambda, \{\text{pk}_i\}_{i=1}^n; r) \\ x \leftarrow \mathcal{E}.\text{Dec}_{\text{sk}_1}(\text{ct}[x]) \end{array} \right],$$

and

$$\Pr[\text{extracted}] = \Pr \left[\mathcal{D}(\mathcal{A}_2(\text{st}, x)) = 1 \left| \begin{array}{l} \text{sk}_i, \text{pk}_i \leftarrow \mathcal{E}.\text{KeyGen}(1^\lambda), \text{ for } i \in [n] \\ \text{ct}[x], \text{st} \leftarrow \mathcal{A}_1^{\text{O}_{\text{En}}, \text{O}_{\text{LR}}}(1^\lambda, \{\text{pk}_i\}_{i=1}^n; r) \\ F \leftarrow \mathcal{A}^*(1^\lambda, \text{'label'}, \{\text{pk}_i\}_{i=1}^n, r; r^*) \\ x_e^{(\ell)} \leftarrow \mathcal{E}.\text{Dec}_{\text{sk}_1}(\text{ct}[x_e^{(\ell)}]), \text{ for } \ell \in [q_{\text{En}}] \\ \vec{\text{ct}}_i^* \leftarrow \mathcal{E}.\text{Dec}_{\text{sk}_1}(\vec{\text{ct}}_i^*), \text{ for } i \in [d+1] \\ x \leftarrow F(\vec{\text{c}}_1, \dots, \vec{\text{c}}_{d+1}, \{x_e^{(\ell)}\}_{\ell=1}^{q_{\text{En}}}) \end{array} \right. \right],$$

is negligible, i.e. $\text{Adv}_{\mathcal{D}}^{\text{mk-ext}}(\lambda) = |\Pr[\text{real}] - \Pr[\text{extracted}]| \leq \text{negl}(\lambda)$, and F is a circuit of depth at most L . The set $\{\text{ct}[x_e^{(\ell)}]\}_{\ell=1}^{q_{\text{En}}}$ denotes the replies of O_{En} , and $\{\vec{\text{ct}}_i^*\}_{i=1}^{d+1}$ are the challenge ciphertexts output by O_{LR} .

- Hybrid $G_2\text{-}0$: This game is identical to G_1 . We now modify the equality oracle in $2(d+1)q$ hybrid steps, enumerating all ciphertexts in the query CT. For notational simplicity, we only show the first two steps.
- Hybrid $G_2\text{-}1$: Instead of decrypting $\text{ct}_{u,1}^{(1)}$, the equality oracle invokes the circuit extractor from Definition 14 as follows:

$$\begin{array}{l} \text{O}_{\text{Equal}}(\text{CT}) \\ \hline 1 : F_1^{(1)} \leftarrow \mathcal{A}^*(\{\text{pk}_{\mathcal{E},i}\}_{i=1}^{d+1}, r, \text{ct}_{u,1}^{(1)}; r^*) \\ 2 : x_e^{(\ell)} \leftarrow \mathcal{E}.\text{Dec}_{\text{sk}_1}(\text{ct}_e^{(j)}), \text{ for } \ell \in [q_{\text{En}}] \\ 3 : \vec{\text{ct}}_i \leftarrow \mathcal{E}.\text{Dec}_{\text{sk}_1}(\vec{\text{ct}}_i^*), \text{ for } i \in [d+1] \\ 4 : u_1^{(1)} \leftarrow F_1^{(1)}(\vec{\text{c}}_1, \dots, \vec{\text{c}}_{d+1}, \{x_e^{(\ell)}\}_{\ell=1}^{q_{\text{En}}}) \\ 5 : u_i^{(j)} \leftarrow \text{Decrypt}(\text{sk}, \text{ct}_{u,i}^{(j)}), \text{ for } (i, j) \in [d+1] \times [q] \setminus \{(1, 1)\} \\ 6 : v_i^{(j)} \leftarrow \text{Decrypt}(\text{sk}, \text{ct}_{v,i}^{(j)}), \text{ for } (i, j) \in [d+1] \times [q] \\ 7 : \textbf{return } 1 \text{ iff. } \bigwedge_{(i,j) \in [d+1] \times [q]} u_i^{(j)} = v_i^{(j)} \end{array}$$

In Line 1, we invoke the circuit extractor to receive $F_1^{(1)}$ which describes $\mathcal{A}$'s computation leading to $\text{ct}_{u,1}^{(1)}$. Since this computation may depend on ciphertexts returned by the encryption oracle O_{En} and the left-or-right oracle O_{LR} , we obtain the corresponding plaintext values in Line 2 and Line 3, respectively. Then, the extracted plaintext $u_1^{(1)}$ is computed via $F_1^{(1)}$ in Line 4.

Note that a distinguisher between $G_2\text{-}0$ and $G_2\text{-}1$ would lead to a distinguisher $\mathcal{D}$ against the circuit extractor $\mathcal{A}^*$, thus adds $\text{Adv}_{\mathcal{D}}^{\text{mk-ext}}(\lambda)$ to the bound.

- Hybrid $G_2\text{-}2$: Instead of decrypting $\text{ct}_{v,1}^{(1)}$, the equality oracle uses the circuit extractor again to obtain $G_1^{(1)}$ which describes $\mathcal{A}$'s computation leading to $\text{ct}_{v,1}^{(1)}$.

$O_{\text{Equal}}(\text{CT})$ in $G_2\text{-}2$

```

1 :  $F_1^{(1)} \leftarrow \mathcal{A}^*(\{\text{pk}_{\mathcal{E},i}\}_{i=1}^{d+1}, r, \text{ct}_{u,1}^{(1)}; r^*)$ 
2 :  $G_1^{(1)} \leftarrow \mathcal{A}^*(\{\text{pk}_{\mathcal{E},i}\}_{i=1}^{d+1}, r, \text{ct}_{v,1}^{(1)}; r^*)$ 
3 :  $x_e^{(\ell)} \leftarrow \mathcal{E}.\text{Dec}_{\text{sk}_1}(\text{ct}_e^{(\ell)})$ , for  $\ell \in [q_{\text{En}}]$ 
4 :  $\hat{\mathcal{C}}_i \leftarrow \mathcal{E}.\text{Dec}_{\text{sk}_1}(\vec{\text{ct}}_i^*)$ , for  $i \in [d+1]$ 
5 :  $u_1^{(1)} \leftarrow F_1^{(1)}(\hat{\mathcal{C}}_1, \dots, \hat{\mathcal{C}}_{d+1}, \{x_e^{(\ell)}\}_{\ell=1}^{q_{\text{En}}})$ 
6 :  $v_1^{(1)} \leftarrow G_1^{(1)}(\hat{\mathcal{C}}_1, \dots, \hat{\mathcal{C}}_{d+1}, \{x_e^{(\ell)}\}_{\ell=1}^{q_{\text{En}}})$ 
7 :  $u_i^{(j)} \leftarrow \text{Decrypt}(\text{sk}, \text{ct}_{u,i}^{(j)})$ , for  $(i, j) \in [d+1] \times [q] \setminus \{(1, 1)\}$ 
8 :  $v_i^{(j)} \leftarrow \text{Decrypt}(\text{sk}, \text{ct}_{v,i}^{(j)})$ , for  $(i, j) \in [d+1] \times [q] \setminus \{(1, 1)\}$ 
9 : return 1 iff.  $\bigwedge_{(i,j) \in [d+1] \times [q]} u_i^{(j)} = v_i^{(j)}$ 

```

Note that $G_1^{(1)}$'s inputs are the same as $F_1^{(1)}$'s since $\mathcal{A}$ has seen the same values when producing it. Again, this step adds $\text{Adv}_{\mathcal{D}}^{\text{mk-ext}}(\lambda)$ to the bound.

- Hybrid $G_2\text{-}2(d+1)q$: In this hybrid, the circuits of all ciphertexts in CT are extracted. The equality oracle now computes:

$O_{\text{Equal}}(\text{CT})$ in $G_2\text{-}2(d+1)q$

```

1 : for  $i \in [d+1]$  do
2 :    $x_e^{(\ell)} \leftarrow \mathcal{E}.\text{Dec}_{\text{sk}_i}(\text{ct}_e^{(\ell)})$ , for  $\ell \in [q_{\text{En}}]$ 
3 :    $\hat{\mathcal{C}}_\iota \leftarrow \mathcal{E}.\text{Dec}_{\text{sk}_i}(\vec{\text{ct}}_\iota^*)$ , for  $\iota \in [d+1]$ 
4 :   for  $j \in [q]$  do
5 :      $F_i^{(j)} \leftarrow \mathcal{A}^*(\{\text{pk}_{\mathcal{E},\iota}\}_{\iota=1}^{d+1}, r, \text{ct}_{u,i}^{(j)}; r^*)$ 
6 :      $G_i^{(j)} \leftarrow \mathcal{A}^*(\{\text{pk}_{\mathcal{E},\iota}\}_{\iota=1}^{d+1}, r, \text{ct}_{v,i}^{(j)}; r^*)$ 
7 :      $u_i^{(j)} \leftarrow F_i^{(j)}(\hat{\mathcal{C}}_1, \dots, \hat{\mathcal{C}}_{d+1}, \{x_e^{(\ell)}\}_{\ell=1}^{q_{\text{En}}})$ 
8 :      $v_i^{(j)} \leftarrow G_i^{(j)}(\hat{\mathcal{C}}_1, \dots, \hat{\mathcal{C}}_{d+1}, \{x_e^{(\ell)}\}_{\ell=1}^{q_{\text{En}}})$ 
9 : return 1 iff.  $\bigwedge_{(i,j) \in [d+1] \times [q]} u_i^{(j)} = v_i^{(j)}$ 

```

- Hybrid G_3 : We continue transforming O_{Equal} . Wlog. we can assume that all $x_e^{(\ell)}$ in Line 2 are wrong-key decryptions. Since if $\text{ct}_e^{(\ell)}$ is encrypted under sk_i , we have

$$x_e^{(\ell)} \leftarrow \mathcal{E}.\text{Dec}_{\text{sk}_i}(\mathcal{E}.\text{Enc}_{\text{sk}_i}(x')) = x',$$

with high probability due to correctness of the HE scheme. Since the adversary supplied x' , it knows $x_e^{(\ell)} = x'$ already. Thus, the extracted $F_i^{(j)}$ can incorporate x' as constant.

Using the multi-message wrong-key decryption property (cf. Definition 4), we replace all wrong-key decryptions of $\text{ct}_e^{(j)}$ with random values.

$\text{O}_{\text{Equal}}(\text{CT})$ in G_3

```

1 : for  $i \in [d+1]$  do
2 :    $x_e^{(\ell)} \leftarrow \mathbb{F}_p$ , for  $\ell \in [q_{\text{En}}]$ 
3 :    $\vec{c}_i \leftarrow \mathcal{E}.\text{Dec}_{\text{sk}_i}(\vec{\text{ct}}_i^*)$ , for  $i \in [d+1]$ 
4 :   for  $j \in [q]$  do
5 :      $F_i^{(j)} \leftarrow \mathcal{A}^*(\{\text{pk}_{\mathcal{E},\ell}\}_{\ell=1}^{d+1}, r, \text{ct}_{u,i}^{(j)}; r^*)$ 
6 :      $G_i^{(j)} \leftarrow \mathcal{A}^*(\{\text{pk}_{\mathcal{E},\ell}\}_{\ell=1}^{d+1}, r, \text{ct}_{v,i}^{(j)}; r^*)$ 
7 :      $u_i^{(j)} \leftarrow F_i^{(j)}(\vec{c}_1, \dots, \vec{c}_{d+1}, \{x_e^{(\ell)}\}_{\ell=1}^{q_{\text{En}}})$ 
8 :      $v_i^{(j)} \leftarrow G_i^{(j)}(\vec{c}_1, \dots, \vec{c}_{d+1}, \{x_e^{(\ell)}\}_{\ell=1}^{q_{\text{En}}})$ 
9 :   return 1 iff.  $\bigwedge_{(i,j) \in [d+1] \times [q]} u_i^{(j)} = v_i^{(j)}$ 

```

The step from G_2 - $2(d+1)q$ to G_3 is indistinguishable since wrong-key decryptions are indistinguishable from random values in the plaintext domain.

- G_4 : We also apply the wrong-key decryption property to $\vec{c}_i$. Indeed, for the i -th component, $\vec{c}_i = \vec{c}_i = \vec{x}_b + \alpha_i \cdot \vec{a}$, while for any $j \neq i$, $\vec{c}_j \leftarrow \mathcal{E}.\text{Dec}_{\text{sk}_i}(\vec{\text{ct}}_j^*)$ is a wrong-key decryption. With similar argument as before, we replace wrong-key decryptions by a random value.

$\text{O}_{\text{Equal}}(\text{CT})$ in G_4

```

1 : for  $i \in [d+1]$  do
2 :    $x_e^{(\ell)} \leftarrow \mathbb{F}_p$ , for  $\ell \in [q_{\text{En}}]$ 
3 :    $\vec{c}_i \leftarrow \vec{x}_b + \alpha_i \cdot \vec{a}$ 
4 :    $\vec{c}_\iota \leftarrow \mathbb{F}_p^\ell$ , for  $\iota \in [d+1] \setminus \{i\}$ 
5 :   for  $j \in [q]$  do
6 :      $F_i^{(j)} \leftarrow \mathcal{A}^*(\{\text{pk}_{\mathcal{E},\ell}\}_{\ell=1}^{d+1}, r, \text{ct}_{u,i}^{(j)}; r^*)$ 
7 :      $G_i^{(j)} \leftarrow \mathcal{A}^*(\{\text{pk}_{\mathcal{E},\ell}\}_{\ell=1}^{d+1}, r, \text{ct}_{v,i}^{(j)}; r^*)$ 
8 :      $u_i^{(j)} \leftarrow F_i^{(j)}(\vec{c}_1, \dots, \vec{c}_{d+1}, \{x_e^{(\ell)}\}_{\ell=1}^{q_{\text{En}}})$ 
9 :      $v_i^{(j)} \leftarrow G_i^{(j)}(\vec{c}_1, \dots, \vec{c}_{d+1}, \{x_e^{(\ell)}\}_{\ell=1}^{q_{\text{En}}})$ 
10 : return 1 iff.  $\bigwedge_{(i,j) \in [d+1] \times [q]} u_i^{(j)} = v_i^{(j)}$ 

```

- Hybrid G_5 : In this hybrid, we change the left-or-right oracle to return encryptions of 0 for each encrypted component $\vec{\text{ct}}_i^*$. Importantly, the oracle O_{Equal} still uses the original encoded challenge $\vec{c}_i = \vec{x}_b + \alpha_i \cdot \vec{a}$ to generate δ .

The step from G_4 to G_5 is indistinguishable up to the advantage $\text{Adv}_{\mathcal{B}}^{\text{ind}}(\lambda)$ of an adversary $\mathcal{B}$ against IND-CPA security of the HE scheme. Since O_{Equal} no longer

decrypts any ciphertext to produce δ , distinguishing between hybrids G_4 and G_5 amounts to distinguishing between $\text{Enc}_{\text{sk}_{\mathcal{E},i}}(\vec{c}_i)$ and $\text{Enc}_{\text{sk}_{\mathcal{E},i}}(\vec{0})$, admitting a reduction to IND-CPA security. Note that in the reduction, whenever there is a query to O_{En} , the reduction forwards the query to the encryption oracle of the IND-CPA game. This step adds $(d+1)\text{Adv}_B^{\text{ind}}(\lambda)$ to the bound.

- Hybrid G_6 : In this step, we modify O_{Equal} by introducing the polynomial $H_i^{(j)}$ defined as

$$\begin{aligned} H_i^{(j)}(\vec{c}_1, \dots, \vec{c}_{d+1}, \{x_e^{(\ell)}\}_{\ell=1}^{q_{\text{En}}}) \\ = F_i^{(j)}(\vec{c}_1, \dots, \vec{c}_{d+1}, \{x_e^{(\ell)}\}_{\ell=1}^{q_{\text{En}}}) - G_i^{(j)}(\vec{c}_1, \dots, \vec{c}_{d+1}, \{x_e^{(\ell)}\}_{\ell=1}^{q_{\text{En}}}). \end{aligned}$$

Since $F_i^{(j)}$ and $G_i^{(j)}$ have bounded degree $\leq 2^L$, $H_i^{(j)}$ also has degree $\leq 2^L$. Further, we now replace the comparison $u_i^{(j)} = v_i^{(j)}$ by $H_i^{(j)}(\vec{c}_1, \dots, \vec{c}_{d+1}, \{x_e^{(\ell)}\}_{\ell=1}^{q_{\text{En}}}) = 0$. The updated oracle is found below.

$\text{O}_{\text{Equal}}(\text{CT})$ in G_6

```

1 : for  $i \in [d+1]$  do
2 :    $x_e^{(\ell)} \leftarrow \mathbb{F}_p$ , for  $\ell \in [q_{\text{En}}]$ 
3 :    $\vec{c}_i \leftarrow \vec{x}_b + \alpha_i \cdot \vec{a}$ 
4 :    $\vec{c}_\iota \leftarrow \mathbb{F}_p^\ell$ , for  $\iota \in [d+1] \setminus \{i\}$ 
5 :   for  $j \in [q]$  do
6 :      $F_i^{(j)} \leftarrow \mathcal{A}^*(\{\text{pk}_{\mathcal{E},\iota}\}_{\iota=1}^{d+1}, r, \text{ct}_{u,i}^{(j)}; r^*)$ 
7 :      $G_i^{(j)} \leftarrow \mathcal{A}^*(\{\text{pk}_{\mathcal{E},\iota}\}_{\iota=1}^{d+1}, r, \text{ct}_{v,i}^{(j)}; r^*)$ 
8 :      $H_i^{(j)} := F_i^{(j)} - G_i^{(j)}$ 
9 : return 1 iff.  $\bigwedge_{(i,j) \in [d+1] \times [q]} H_i^{(j)}(\vec{c}_1, \dots, \vec{c}_{d+1}, \{x_e^{(\ell)}\}_{\ell=1}^{q_{\text{En}}}) = 0$ 

```

This change is syntactical. Indeed, if $u_i^{(j)} = v_i^{(j)}$, then $u_i^{(j)} - v_i^{(j)} = 0$. Thus, δ is identically distributed in both games, and the advantage remains the same.

Note now that the adversary's query CT to O_{Equal} is independent from the secret bit b since it does not learn any values derived from b before making the query. Recall that the challenge ciphertexts $\vec{\text{ct}}_i^*$ are encryptions of 0, thus are independent of b . Only the response $\delta \leftarrow \text{O}_{\text{Equal}}(\text{CT})$ depends on b .

Making use of this fact, the following lemma shows that the advantage in G_6 is bounded by the distribution of δ .

Lemma 1. *The advantage in G_6 is*

$$\begin{aligned} \text{Adv}_{\mathcal{A}}^{G_6}(\lambda) &= |\Pr[\mathcal{A}_2(\text{st}_{\mathcal{A}}, \delta) = 1 | b = 1] - \Pr[\mathcal{A}_2(\text{st}_{\mathcal{A}}, \delta) = 1 | b = 0]| \\ &\leq |\Pr[\delta = 1 | b = 1] - \Pr[\delta = 1 | b = 0]|. \end{aligned}$$

Proof. Denote with $A(\delta)$ the event that $\mathcal{A}_2(\text{st}_{\mathcal{A}}, \delta) = 1$, since for a fixed δ , $\Pr[\mathcal{A}_2(\text{st}_{\mathcal{A}}, \delta) = 1 | b = 0] = \Pr[\mathcal{A}_2(\text{st}_{\mathcal{A}}, \delta) = 1 | b = 1]$ as δ is the only value in the game that depends on b . Further, denote with Δ_β the event that $\delta = 1 | b = \beta$. For $\beta \in \{0, 1\}$, write

$$\begin{aligned}\Pr[A(\delta) | b = \beta] &= \Pr[A(1) | b = \beta] \cdot \Pr[\Delta_\beta] + \Pr[A(0) | b = \beta] \cdot (1 - \Pr[\Delta_\beta]) \\ &= \Pr[A(1)] \cdot \Pr[\Delta_\beta] + \Pr[A(0)] \cdot (1 - \Pr[\Delta_\beta])\end{aligned}$$

Then,

$$\begin{aligned}|\Pr[A(\delta) | b = 1] - \Pr[A(\delta) | b = 0]| &= |(\Pr[\Delta_1] - \Pr[\Delta_0]) \cdot (\Pr[A(1)] - \Pr[A(0)])| \\ &\leq |\Pr[\Delta_1] - \Pr[\Delta_0]| \quad (\text{as } |\Pr[A(1)] - \Pr[A(0)]| \leq 1),\end{aligned}$$

as desired.

We continue with the main proof. Consider two cases.

Case 1: All $H_i^{(j)}$ are degenerate. Assume that for all $i \in [d+1]$ and $j \in [q]$, $H_i^{(j)}$ is the zero polynomial. Then, O_{Equal} always returns $\delta = 1$, and thus there is no advantage:

$$\text{Adv}_{\mathcal{A}}^{G_6}(\lambda) \leq |\Pr[\delta = 1 | b = 1] - \Pr[\delta = 1 | b = 0]| = |1 - 1| = 0.$$

Case 2: Not all $H_i^{(j)}$ are degenerate. This means that there exists at least one $H_{i^*}^{(j^*)}$ that is not the zero polynomial. Note that, for any $\beta \in \{0, 1\}$,

$$\Pr[\delta = 1 | b = \beta] \leq \Pr\left[H_{i^*}^{(j^*)}(\hat{c}_1, \dots, \hat{c}_{d+1}, \{x_e^{(\ell)}\}_{\ell=1}^{q_{\text{En}}}) = 0 | b = \beta\right],$$

since $(\hat{c}_1, \dots, \hat{c}_{d+1}, \{x_e^{(\ell)}\}_{\ell=1}^{q_{\text{En}}})$ must be roots of $H_{i^*}^{(j^*)}$ for the oracle to return $\delta = 1$. Additional conditions, i.e. being roots of more polynomials $H_i^{(j)}$, can only decrease the probability. We will now drop those indices and just write H instead of $H_{i^*}^{(j^*)}$.

- Hybrid G_7 : This hybrid is identical to G_6 except that in O_{Equal} , the oracle now also samples a random value $\hat{c}_{i^*} \leftarrow \mathbb{F}_p^\ell$. Then with the concatenation of all inputs $\vec{r} = (\hat{c}_1, \dots, \hat{c}_{d+1}, \{x_e^{(t)}\}_{t=1}^{q_{\text{En}}})$ of length $(d+1)\ell + q_{\text{En}}$, we have

$$H(\vec{r}) = 0.$$

This change is indistinguishable since $\vec{x}_b + \alpha_{i^*} \cdot \vec{a} \approx \hat{c}_{i^*}$ (for uniform random $\hat{c}_{i^*}$) since $\vec{a}$ is uniformly random and $\alpha_{i^*} \neq 0$.

Then for any $\beta \in \{0, 1\}$,

$$\Pr[H(\vec{r}) = 0 | b = \beta] \leq \frac{2^L}{p},$$

by the Schwartz-Zippel lemma since H and $\vec{r}$ are chosen independently from each other and from b , and $\vec{r}$ is uniformly random. Similar to Lemma 1, we have

$$\text{Adv}_{\mathcal{A}}^{G_7}(\lambda) \leq |\Pr[H(\vec{r}) = 0 \mid b = 1] - \Pr[H(\vec{r}) = 0 \mid b = 0]| \leq \frac{2^L}{p}.$$

This leads to the final bound and concludes the proof. $\square$

C Verifiable homomorphic matrix vector multiplication

We first present two VC protocols for plaintext matrix vector multiplication in Appendix C.1. Then we construct vCOED schemes with strong client privacy for homomorphic multiplications between plaintext matrices and ciphertext vectors, following the approach outlined in Section 3.2 (Fig. 3). Specifically, we combine the first VC protocol with HERE for linear functions in Fig. 4 to construct the vCOED scheme vCOEDMVI in Appendix C.2, and the second VC protocol with the same HERE in Fig. 4 to construct another vCOED scheme, vCOEDMVI, in Appendix C.3.

C.1 VC protocols for plaintext matrix vector multiplication

In this subsection, we introduce protocols that *only* provide integrity for matrix vector multiplications *without* privacy. Consider a client who holds $\vec{u} \in \mathbb{F}^n$ and wants to obtain the matrix-vector product $\vec{v} = \mathbf{A}\vec{u} \in \mathbb{F}^m$ for a large public matrix $\mathbf{A} \in \mathbb{F}^{m \times n}$. The client receives a vector $\vec{v} \in \mathbb{F}^m$ from the server and aims to efficiently verify whether $\vec{v}$ is equal to the desired output $\vec{v} = \mathbf{A}\vec{u}$. This naturally leads us to verifiable computation (VC) protocols.

We first present a classic VC protocol [11, 12, 75], where the prover performs no additional work beyond computing the matrix vector product. This efficiency comes at the cost of a one-time $\mathcal{O}((m \cdot n))$ preprocessing phase for the verifier, where the client processes the entire matrix $\mathbf{A}$ to generate a verification key. The resulting verification key is private and tied to the verifier: it can be reused across multiple computations with respect to the same matrix $\mathbf{A}$, but cannot be made public. Revealing it compromises soundness.

We then introduce a second protocol that combines the multivariate sum-check protocol with a polynomial commitment scheme (PCS). The resulting SNARK protocol is similar to the lincheck step of the Spartan [69] SNARK except it is tailored to general matrix vector multiplications. Compared to the first protocol, this approach introduces additional prover costs, but it does not require expensive verifier preprocessing. Instead, any trusted party (including the publisher of $\mathbf{A}$) can publish a commitment to $\mathbf{A}$, which enables a public verification.

Protocol I: Randomised linear combination check. During preprocessing, the verifier samples a random row vector $\mathbf{r} \leftarrow \mathbb{F}^m$, computes $\mathbf{s} = \mathbf{r} \cdot \mathbf{A} \in \mathbb{F}^n$ (requiring $\mathcal{O}((m \cdot n))$ cost), and stores $(\mathbf{r}, \mathbf{s})$ as a private verification key.

For any input $\vec{u}$, the output $\vec{v}$ obtained via delegation is accepted if

$$\mathbf{s} \cdot \vec{u} = \mathbf{r} \cdot \vec{v}. \quad (3)$$

The verification cost is $\mathcal{O}((m+n))$, which is cheaper than directly computing $\mathbf{A}\vec{u}$. Although the preprocessing cost incurs a one-time cost of $\mathcal{O}((m \cdot n))$, the generated verification key can be reused across multiple matrix vector multiplications with respect to the same $\mathbf{A}$, assuming the verification key is kept secret from the server.

The soundness of this protocol follows from the Schwartz-Zippel lemma. Consider a non-zero m -variate polynomial over $\mathbb{F}$

$$g(x_0, \dots, x_{m-1}) = \sum_{i=0}^{m-1} (\vec{v}[i] - \hat{\vec{v}}[i]) \cdot x_i,$$

then we have $\Pr_{\mathbf{r} \leftarrow \mathbb{F}^m} [g(\mathbf{r}) = 0] \leq \frac{1}{|\mathbb{F}|}$. Therefore, for any incorrect output $\vec{v} \neq \vec{v}$, the verification in Equation (3) passes with probability at most $\frac{1}{|\mathbb{F}|}$, assuming that the prover does not learn the private verification key $(\mathbf{r}, \mathbf{s})$. We emphasize that the secrecy of the verification key is crucial: if the server learns it, it can trivially construct $\vec{v}$ that satisfies Equation (3), thus breaking soundness.

Put another way, soundness relies on the difficulty of finding a non-zero error vector $\vec{e}_o$ such that $\mathbf{r} \cdot \vec{e}_o = 0$. For a fresh random $\mathbf{r}$, a non-zero $\vec{e}_o$ is accepted with negligible probability ($\leq \frac{1}{|\mathbb{F}|}$). However, when the same verification key is reused across multiple multiplications with respect to $\mathbf{A}$, a malicious server may adaptively exploit verification outcomes to gain information about $\mathbf{r}$. In particular, each verification for a malicious output reveals whether $\mathbf{r} \cdot \vec{e} = 0$ for some chosen $\vec{e} \neq 0$. Therefore, to mitigate such leakages, we require the client to rerun the preprocessing phase after any verification failure.

Finally, if a malicious output is accepted (which happens with negligible probability), then the server finds a non-zero vector $\vec{e}_o$ such that $\mathbf{r} \cdot \vec{e}_o = 0$, which can be reused to forge outputs of the form $\vec{v} = \mathbf{A}\vec{u} - \vec{e}_o$, breaking soundness of all subsequent multiplications with respect to $\mathbf{A}$. This is analogous to VC schemes with a trusted setup (e.g. Pinocchio [66]): if the underlying hardness assumption is violated (which happens with negligible probability), then all subsequent computations lose their soundness guarantees.

Protocol II: Sumcheck-based SNARK. Let $\mathbf{A} \in \mathbb{F}^{m \times n}$ denote a matrix. We can equivalently view it as a function $\{0, 1\}^\mu \times \{0, 1\}^\nu \rightarrow \mathbb{F}$ (where $\mu = \lceil \log m \rceil$ and $\nu = \lceil \log n \rceil$) that maps two indices of μ bits and ν bits to the corresponding entry of $\mathbf{A}$. Similarly, let $\vec{u} \in \mathbb{F}^n$, $\vec{v} \in \mathbb{F}^m$ be two column vectors, viewed as functions $\{0, 1\}^\nu \rightarrow \mathbb{F}$ and $\{0, 1\}^\mu \rightarrow \mathbb{F}$ respectively.

During preprocessing, any trusted party (in particular the publisher of the matrix $\mathbf{A}$) can generate and publish a commitment $C_{\mathbf{A}}$ to the multilinear polynomial $\mathbf{A} \in \mathbb{F}[X_1, \dots, X_{\mu+\nu}]$, using a transparent PCS protocol $\mathcal{PC}$ such as BaseFold [74] and Brakedown [43]. This commitment $C_{\mathbf{A}}$ is shared across all

verifiers that check matrix-vector multiplications with respect to the matrix $\mathbf{A}$. Since both $\mathbf{A}$ and $C_{\mathbf{A}}$ are public, any verifier can check the honest generation of $C_{\mathbf{A}}$ using $\text{Open}(\text{pp}, C_{\mathbf{A}}, \mathbf{A})$.

To prove $\vec{v} = \mathbf{A}\vec{u}$, we consider the following function $f : \{0, 1\}^\mu \rightarrow \mathbb{F}$

$$f(\mathbf{x}) = \sum_{\mathbf{y} \in \{0, 1\}^\nu} \mathbf{A}(\mathbf{x}, \mathbf{y}) \cdot u(\mathbf{y}) - v(\mathbf{x}),$$

which evaluates to zero at all points in the μ -dimensional hypercube if and only if $\vec{v} = \mathbf{A}\vec{u}$. Define its low-degree extension $F : \mathbb{F}^\mu \rightarrow \mathbb{F}$ as

$$F(\mathbf{x}) = \sum_{\mathbf{y} \in \{0, 1\}^\nu} \tilde{\mathbf{A}}(\mathbf{x}, \mathbf{y}) \cdot \tilde{u}(\mathbf{y}) - \tilde{v}(\mathbf{x}).$$

Then F is the zero polynomial if and only if f is the zero function over $\{0, 1\}^\mu$.

To prove $F(\cdot)$ is the zero polynomial, it suffices to show that $F(\boldsymbol{\tau}) = 0$ for a random point $\boldsymbol{\tau} \leftarrow \mathbb{F}^\mu$, which incurs a soundness error $\delta_s \leq \mu/|\mathbb{F}|$. For a fixed $\boldsymbol{\tau}$, define $\mathcal{G}(\mathbf{y}) = \tilde{\mathbf{A}}(\boldsymbol{\tau}, \mathbf{y}) \cdot \tilde{u}(\mathbf{y})$, a ν -variate polynomial with degree at most $\ell = 2$ in each variable. Then proving $F(\boldsymbol{\tau}) = 0$ is equivalent to proving

$$\tilde{v}(\boldsymbol{\tau}) = \sum_{\mathbf{y} \in \{0, 1\}^\nu} \mathcal{G}(\mathbf{y}). \quad (4)$$

The verifier can efficiently compute $\tilde{v}(\boldsymbol{\tau})$. Using the sum-check protocol, verifying Eq. (4) is reduced to checking a single evaluation of $\mathcal{G}$ at a random point $\mathbf{r}$. Concretely, $\mathbf{V}$ computes $\mathcal{G}(\mathbf{r}) = \tilde{\mathbf{A}}(\boldsymbol{\tau}, \mathbf{r}) \cdot \tilde{u}(\mathbf{r})$ by (i) obtaining $\tilde{\mathbf{A}}(\boldsymbol{\tau}, \mathbf{r})$ via the PCS evaluation protocol $\mathcal{PC}.\text{Eval}$, and (ii) deriving $\tilde{u}(\mathbf{r})$ from $\vec{u}$. Note $\mathbf{V}$ verifies only using $C_{\mathbf{A}}$ (without access to $\mathbf{A}$), achieving succinct verification. The full protocol is described in Fig. 6.

In the following vCOED protocols in Appendices C.2 and C.3, the evaluation points $\alpha_i = i$ are fixed, public protocol parameters (as in Fig. 4); since our encoding has no encoding secret key ($\text{sk}_1 = \emptyset$), α_1, α_2 are simply reused across all ProbGen queries and do not need to be sampled or kept secret.

C.2 vCOED built from HERE and VC protocol I

We build vCOEDMVI, a vCOED protocol for matrix-vector multiplication from combining HERE and the VC protocol I described in Appendix C.1.

Setup: Given a security parameter λ and a public matrix $\mathbf{A} \in \mathbb{F}^{m \times n}$, the protocol contains two subroutines:

- **KeyGen** in a levelled HE scheme $\mathcal{E} = (\mathcal{E}.\text{KeyGen}, \mathcal{E}.\text{Enc}, \mathcal{E}.\text{Dec}, \mathcal{E}.\text{Eval})$ with plaintext space $\mathbb{F}$. In practice, the plaintext space may be $\mathbb{F}^\ell$ to support ℓ SIMD slots, but here we ignore this distinction for simplicity. Our vCOED protocol can also be instantiated using additive HE schemes. Run $(\text{sk}_{\mathcal{E}, i}, \text{pk}_{\mathcal{E}, i}) \leftarrow \mathcal{E}.\text{KeyGen}(1^\lambda)$ for $i = 1, 2$.

- **Setup** in VC protocol I. Concretely, sample a random row vector $r \leftarrow \mathbb{F}^m$, compute $s = r \cdot A \in \mathbb{F}^n$, and set $vk = (r, s)$.

Finally, it outputs $sk = (sk_{\mathcal{E},1}, sk_{\mathcal{E},2}, vk)$ and $pk = (pk_{\mathcal{E},1}, pk_{\mathcal{E},2})$.

ProbGen: Let $\alpha_i = i$. Given the secret key sk from **Setup**, for any input vector $\vec{u} \in \mathbb{F}^n$, the protocol computes

1. **Encode**($\vec{u}$): sample $\vec{a} \leftarrow \mathbb{F}^n$, compute $\vec{c}_{u,1} := \vec{u} + \alpha_1 \cdot \vec{a}$ and $\vec{c}_{u,2} := \vec{u} + \alpha_2 \cdot \vec{a}$.
2. **Encrypt**(($sk_{\mathcal{E},1}, sk_{\mathcal{E},2}$), ($\vec{c}_{u,1}, \vec{c}_{u,2}$)): encrypt $\vec{ct}_{u,1} \leftarrow \mathcal{E}.\text{Enc}(sk_{\mathcal{E},1}, \vec{c}_{u,1})$ and $\vec{ct}_{u,2} \leftarrow \mathcal{E}.\text{Enc}(sk_{\mathcal{E},2}, \vec{c}_{u,2})$.

Output $\text{Enc}(\vec{u}) = (\vec{ct}_{u,1}, \vec{ct}_{u,2})$ and the client state $st = (\vec{c}_{u,1}, \vec{c}_{u,2})$.

Compute: On input $\text{Enc}(\vec{u}) = (\vec{ct}_{u,1}, \vec{ct}_{u,2})$, perform the following homomorphic evaluations:

- Compute $\vec{ct}_{v,1} \leftarrow \mathcal{E}.\text{Eval}_{pk_{\mathcal{E},1}}(A, \vec{ct}_{u,1})$
- Compute $\vec{ct}_{v,2} \leftarrow \mathcal{E}.\text{Eval}_{pk_{\mathcal{E},2}}(A, \vec{ct}_{u,2})$

where $\mathcal{E}.\text{Eval}(A, \cdot)$ denotes the homomorphic evaluation of PC-MV. Output $\text{Enc}(\vec{v}) = (\vec{ct}_{v,1}, \vec{ct}_{v,2})$.

Verify: On input $\text{Enc}(\vec{v}) = (\vec{ct}_{v,1}, \vec{ct}_{v,2})$, the protocol uses $sk = (sk_{\mathcal{E},1}, sk_{\mathcal{E},2}, vk)$, the client state $st = (\vec{c}_{u,1}, \vec{c}_{u,2})$, and computes the following:

1. Decrypt $\vec{c}_{v,1} = \mathcal{E}.\text{Dec}(sk_{\mathcal{E},1}, \vec{ct}_{v,1})$ and $\vec{c}_{v,2} = \mathcal{E}.\text{Dec}(sk_{\mathcal{E},2}, \vec{ct}_{v,2})$
2. Using the verification key $vk = (r, s)$, check if $s \cdot \vec{c}_{u,1} \stackrel{?}{=} r \cdot \vec{c}_{v,1}$ and $s \cdot \vec{c}_{u,2} \stackrel{?}{=} r \cdot \vec{c}_{v,2}$ both hold. If either check fails, output **rej**.
3. If both checks pass, decode by computing $\vec{v} = L_1 \cdot \vec{c}_{v,1} + L_2 \cdot \vec{c}_{v,2}$, where L_i denote the Lagrange interpolator of point α_i , interpolating at 0.

C.3 vCOED built from HERE and VC protocol II

We build vCOEDMVII, a vCOED protocol for matrix-vector multiplication from combining HERE and the VC protocol II described in Appendix C.1.

Setup: Given a security parameter λ and a public matrix $A \in \mathbb{F}^{m \times n}$, the protocol contains two subroutines:

- **KeyGen** in a levelled HE scheme $\mathcal{E} = (\mathcal{E}.\text{KeyGen}, \mathcal{E}.\text{Enc}, \mathcal{E}.\text{Dec}, \mathcal{E}.\text{Eval})$ with plaintext space $\mathbb{F}$. Our vCOED protocol can also be instantiated using additive HE schemes. Run $(sk_{\mathcal{E},i}, pk_{\mathcal{E},i}) \leftarrow \mathcal{E}.\text{KeyGen}(1^\lambda)$ for $i = 1, 2$.
- **Setup** in VC protocol II. Concretely, view the matrix A as a multilinear polynomial whose evaluations are the corresponding elements, and generate a commitment C_A using a transparent PCS protocol. Set $ek = C_A$.

Finally, it outputs $sk = (sk_{\mathcal{E},1}, sk_{\mathcal{E},2})$ and $pk = (pk_{\mathcal{E},1}, pk_{\mathcal{E},2}, ek)$.

ProbGen: Let $\alpha_i = i$. Given the secret key sk from **Setup**, for any input vector $\vec{u} \in \mathbb{F}^n$, the protocol computes

1. **Encode**($\vec{u}$): sample $\vec{a} \leftarrow \mathbb{F}^n$, compute $\vec{c}_{u,1} := \vec{u} + \alpha_1 \cdot \vec{a}$ and $\vec{c}_{u,2} := \vec{u} + \alpha_2 \cdot \vec{a}$.
2. **Encrypt**(($\text{sk}_{\mathcal{E},1}, \text{sk}_{\mathcal{E},2}$), ($\vec{c}_{u,1}, \vec{c}_{u,2}$)): encrypt $\vec{\text{ct}}_{u,1} \leftarrow \mathcal{E}.\text{Enc}(\text{sk}_{\mathcal{E},1}, \vec{c}_{u,1})$ and $\vec{\text{ct}}_{u,2} \leftarrow \mathcal{E}.\text{Enc}(\text{sk}_{\mathcal{E},2}, \vec{c}_{u,2})$.

Output $\text{Enc}(\vec{u}) = (\vec{\text{ct}}_{u,1}, \vec{\text{ct}}_{u,2})$ and the client state $\text{st} = (\vec{c}_{u,1}, \vec{c}_{u,2})$.

Compute: On input $\text{Enc}(\vec{u}) = (\vec{\text{ct}}_{u,1}, \vec{\text{ct}}_{u,2})$, compute $\vec{\text{ct}}_{v,i} \leftarrow \mathcal{E}.\text{Eval}_{\text{pk}}(\mathbf{A}, \vec{\text{ct}}_{u,i})$ for $i = 1, 2$. Then homomorphically evaluate the non-interactive variant of the proving procedure in Fig. 6 to obtain ct_π . This evaluation follows the blind zkSNARK paradigm from [34], where challenges are derived from ciphertexts. Output $\text{Enc}(\vec{v}) = (\vec{\text{ct}}_{v,1}, \vec{\text{ct}}_{v,2})$ and $\text{ct}_\pi = (\text{ct}_{\pi_1}, \text{ct}_{\pi_2})$.

Verify: On input $\text{Enc}(\vec{v}) = (\vec{\text{ct}}_{v,1}, \vec{\text{ct}}_{v,2})$ and ct_π , the protocol uses $\text{sk} = (\text{sk}_{\mathcal{E},1}, \text{sk}_{\mathcal{E},2})$, the commitment ek , and the client state $\text{st} = (\vec{c}_{u,1}, \vec{c}_{u,2})$, and computes the following:

1. Decrypt $\vec{c}_{v,i} = \mathcal{E}.\text{Dec}(\text{sk}_{\mathcal{E},i}, \vec{\text{ct}}_{v,i})$ and $\pi_i = \mathcal{E}.\text{Dec}(\text{sk}_{\mathcal{E},i}, \text{ct}_{\pi_i})$ for $i = 1, 2$.
2. Run the SNARK verification as in Fig. 6, which takes (π_1, ek) and (π_2, ek) as separate inputs, and check that all the challenges are correctly derived from committed ciphertexts. Output **rej** if any check fails.
3. If the verification is not rejected, decode by computing $\vec{v} = L_1 \cdot \vec{c}_{v,1} + L_2 \cdot \vec{c}_{v,2}$, where L_i denote the Lagrange interpolator of point α_i , interpolating at 0.

C.4 Our combined vCOED protocol

Our third vCOED construction for matrix-vector multiplication combines the advantages of the previous two approaches.

The main limitation of vCOEDMVI is the expensive **Setup** in VC protocol I: the client samples a random row vector $r \leftarrow \mathbb{F}^m$, computes $s = r \cdot \mathbf{A} \in \mathbb{F}^n$, and sets the verification key as $\text{vk} = (r, s)$, which must be kept secret from the server for soundness. This requires the client to download $\mathbf{A}$ and compute $s = r \cdot \mathbf{A}$, which may be prohibitively expensive for resource-constrained clients.

In the second approach vCOEDMVII, the commitment $C_{\mathbf{A}}$ can be published together with the matrix $\mathbf{A}$, which allows public verification and can be reused when verifying all queries from different clients. However, this approach incurs larger proof overhead, with server computation of $\mathcal{O}(mn \log(mn))$ compared to $\mathcal{O}(mn)$ in vCOEDMVI.

Our combined protocol draws on both approaches. The online phase (**ProbGen**, **Compute**, and **Verify**) are the same as vCOEDMVI, maintaining $\mathcal{O}(mn)$ server computation cost. The key innovation lies in the one-time offline setup: the client securely and verifiably delegates **KeyGen** in VC protocol I to the server using vCOEDMVII. Specifically, the client samples $r \leftarrow \mathbb{F}^m$, and runs vCOEDMVII with the server to obtain $\vec{s} := \mathbf{A}^\top \cdot \vec{r}$ for $\vec{r} = r^\top$, and the commitment $C_{\mathbf{A}^\top}$ is a public input. The client then sets $s = \vec{s}^\top$ and $\text{vk} = (r, s)$. Crucially, the derived verification key vk remains secret from the server while enabling verification of all subsequent online queries.

Proving matrix vector multiplication via the sum-check based protocol

The first round

- P sends the claimed sum $\tilde{v}(\tau)$ as well as a univariate polynomial $\mathcal{G}_1(Y)$ (in the form of evaluations $\{\mathcal{G}_1(0), \mathcal{G}_1(1), \mathcal{G}_1(2)\}$), where $\mathcal{G}_1(Y)$ is claimed to be

$$\sum_{(y_2, \dots, y_\nu) \in \{0,1\}^{\nu-1}} \mathcal{G}(Y, y_2, \dots, y_\nu).$$

V checks $\tilde{v}(\tau) = \mathcal{G}_1(0) + \mathcal{G}_1(1)$ and rejects if not satisfied.

- V chooses a random element $r_1 \in \mathbb{F}$ and sends r_1 to P.

The j -th round, for $1 < j < \nu$

- P sends to V a univariate polynomial $\mathcal{G}_j(Y)$ (in the evaluation form) claimed to equal

$$\sum_{(y_{j+1}, \dots, y_\nu) \in \{0,1\}^{\nu-j}} \mathcal{G}(r_1, \dots, r_{j-1}, Y, y_{j+1}, \dots, y_\nu).$$

V checks that $\mathcal{G}_{j-1}(r_{j-1}) = \mathcal{G}_j(0) + \mathcal{G}_j(1)$, where the left-hand side is derived from point values received from the prover in the previous round. If the check fails, V rejects.

- V chooses a random element $r_j \in \mathbb{F}$ and sends r_j to P.

The final round

- P sends to V a univariate polynomial (in the evaluation form) $\mathcal{G}_\nu(Y)$ claimed to equal

$$\mathcal{G}(r_1, \dots, r_{\nu-1}, Y).$$

V checks that $\mathcal{G}_{\nu-1}(r_{\nu-1}) = \mathcal{G}_\nu(0) + \mathcal{G}_\nu(1)$ and rejects if not satisfied.

- V chooses a random element $r_\nu \in \mathbb{F}$. Let $\mathbf{r} = (r_1, \dots, r_\nu)$. V computes $\mathcal{G}(\mathbf{r}) = \tilde{\mathbf{A}}(\tau, \mathbf{r}) \cdot \tilde{\mathbf{u}}(\mathbf{r})$ by (i) obtaining $\tilde{\mathbf{A}}(\tau, \mathbf{r})$ from the server, checking its correctness with respect to the commitment $C_{\mathbf{A}}$ via the PCS evaluation protocol $\mathcal{PC}.\text{Eval}$, and (ii) deriving $\tilde{\mathbf{u}}(\mathbf{r})$ from $\tilde{\mathbf{u}}$. Finally, V checks that

$$\mathcal{G}_\nu(r_\nu) = \mathcal{G}(\mathbf{r}),$$

and rejects if not satisfied.

- V accepts if all checks pass.

Fig. 6: Proving matrix vector multiplication $\vec{v} = \mathbf{A}\vec{u}$ using the sum-check based protocol (Protocol II, Section C.1). Both P and V hold $\vec{u}$ and $\vec{v}$. Additionally, P knows $\mathbf{A}$; V verifies only using $C_{\mathbf{A}}$ (without access to $\mathbf{A}$), achieving succinct verification.